**BitTorrent network**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**מגיש: אימרי הלל**

**ת"ז: 215428707**

**מורה: מרי גבע**

**בית הספר: כרמים בנימינה**

**תאריך 12.5.24**

תוכן עניינים

[פרק א - nexus- ייזום 4](#_Toc166419718)

[תיאור ראשוני של המערכת 4](#_Toc166419719)

[פרק ב' - nexus - אפיון 6](#_Toc166419720)

[פרוט המערכת: 6](#_Toc166419721)

[פירוט הבדיקות ('קופסא שחורה') 6](#_Toc166419722)

[תכנון לוח זמנים לפרויקט 8](#_Toc166419723)

[ניהול סיכונים בפרויקט 9](#_Toc166419724)

[פרק ג' - Nexus- מסמך ניתוח 10](#_Toc166419725)

[פרוט יכולות המערכת 10](#_Toc166419726)

[פרק ד' - Nexus - העיצוב 15](#_Toc166419727)

[תיאור הארכיטקטורה של המערכת המוצעת 15](#_Toc166419728)

[תיאור הטכנולוגיה הרלוונטית 15](#_Toc166419729)

[מודולים בשרת 23](#_Toc166419730)

[מודולים בלקוח 28](#_Toc166419731)

[מודולים משותפים 34](#_Toc166419732)

[תיאור האלגוריתמים המרכזיים בפרויקט: 38](#_Toc166419733)

[תיאור מסכי הפרויקט: 41](#_Toc166419734)

[תיאור פרוטוקול התקשורת 42](#_Toc166419735)

[תיאור מבני הנתונים 44](#_Toc166419736)

[סקירת חולשות והאיומים 45](#_Toc166419737)

[פרק ה' - nexus – הקוד 46](#_Toc166419738)

[קטעי קוד מעניינים 46](#_Toc166419739)

[פרק ו' – nexus – בדיקות ('קופסא לבנה') 49](#_Toc166419740)

[פירוט הבדיקות: 49](#_Toc166419741)

[פרק ז' - nexus – מדריך למשתמש 53](#_Toc166419742)

[מדריך למשתמש (בוחן) 53](#_Toc166419743)

[מדריך למשתמש 54](#_Toc166419744)

[פרק ח' - nexus – רפלקציה 57](#_Toc166419745)

[מבט אישי על העבודה ועל תהליך פיתוחה: 57](#_Toc166419746)

[פרק ט' - nexus – ביבליוגרפיה 58](#_Toc166419747)

## פרק א - nexus- ייזום

### תיאור ראשוני של המערכת

* תקציר כולל ורציונל הפרויקט - נקסוס היא רשת של לקוחות שמוכנים ומעוניינים לתקשר אחד עם השני בשביל האופציה להוריד קבצים אחד מהשני בשביל מהירות הורדה גבוהה יותר .
* מה המוצר המוגמר אמור לבצע – המוצר הסופי אמור לאפשר ללקוחות להוריד כל קובץ שקיים לפחות פעם אחת ברשת אצל לקוח אחר והורדה מהירה יותר של קבצים שקיימים יותר מפעם אחת ברשת.
* למה בחרתי בפרויקט ומה האתגרים שאני צופה לי בפרויקט – בחרתי בפרויקט משום שהתחברתי מאוד לרעיון עצמו ובנוסף לזה אני מרגיש שהפרויקט מאוד מתקשר לחיי היום יום שלי ואני חושב שפרויקט כזה יכול להיות שימושי להרבה מאוד אנשים. אחד האתגרים שצפויים לי בפרויקט הוא להגיע לשליטה עמוקה בפורמט של הקובץ ביטורנט. אתגר נוסף שיכול להיות לי הוא להבין איך לממש את הורדת הקבצים ברקע של התוכנית מבלי להפריע לפעילות הלקוח במחשב.

1. **הגדרת הלקוח**

* למי מיועדת המערכת ומי הולך להשתמש בה? – הפרויקט מיועד לכל לקוח שמעוניין לקצר את זמני ההורדה של הקבצים שלו, כגון לקוחות שצופים בסרטים ארוכים או לקוחות שמעוניינים להוריד קבצים חדשים שיצאו לאחרונה ואינם מעוניינים להוריד אותם מהמקור שלהם כדי לחסוך בזמן ההורדה.

1. **הגדרת יעדים/מטרות**

* מה המטרות המרכזיות של המערכת המוצעת (לפרט לכל לקוח ולקוח) – בראיית הלקוח.

מטרת כל לקוח היא לאפשר הורדת קבצים מהירה יותר ללקוחות אחרים ולשיפור מהירות הורדת הקבצים ברשת בנוסף לזה המטרה האישית של כל לקוח היא להוריד בעצמו קבצים יותר מהר. ומטרה נוספת שמאוד חשובה היא להבטיח ללקוחות שמורידים קבצים דרך המערכת דרכים לאמת את שלמות ומהימנות המידע המורד דרך המערכת.

1. **בעיות, תועלות וחסכונות**

* מה הבעיה/יות ומה אנחנו מנסים להשיג? – בכלליות הבעיה היא שהורדת קבצים היא לא מספיק אופטימלית ויכולה להיות יותר מהירה על ידי הורדה ממקומות שונים. אנחנו מנסים להשיג הורדת קבצים מהירה יותר ויעילה יותר
* מה התועלות שסביר לצפות מהמערכת? – בסיום הפרויקט קבצים יוכלו לרדת הרבה יותר מהר (במידה והקבצים קיימים כמה פעמיים ברשת) ובמידה ולקוח ירצה שיורידו ממנו קובץ חדש לדוגמא סרט שהוא הכין או כל תוצר אחר שהביקוש שלו יכול להיות גבוהה התוצר יאפשר הורדת עומס מאותו משתמש

* אלו שירותים המערכת תיתן? – המערכת תיתן שירות אחד שהוא הורדת קבצים שקיימים ברשת במהירות גבוהה מהרגיל תוך מתן ערובה למהימנות ושלמות הקבצים המורדים.
* השוואת העבודה עם פתרונות ויישומים קיימים– יישומים שקיימים ופותרים את הבעיה הם bit torrent network , internet Download Manager ההבדל הוא שאת הפרויקט אני מממש ולא משתמש בתוצר קיים. מה שמאפשר לי חופש עבודה והוספת יכולות שאני מעוניין להוסיף.

1. **האם צפויים קשיים או מגבלות בהגדרת המערכת**

* האם מדובר בטכנולוגיה חדשה ובלתי מוכרת? – לא מדובר בטכנולוגיה חדשה או מסוכנת אשתמש ברשת תקשורת, קבצים, אממש תהליכים של העלאה והורדה כולל אלמנטים של אבטחת מידע.
* האם קיימים סייגים בהגדרת המערכת?– למימוש אין דרישות של ציוד ספציפי בגלל המגבלה של רשת התקשורת הסגורה בבית הספר – רק משתמשים ברשת הזו יוכלו להשתמש במערכת.

1. **תיחום הפרויקט**

* הפרויקט עוסק בתחומים רבים: בתחום הרשתות במידע שעובר ברשת, בתחום מערכות ההפעלה הפעלת טרדים רבים, תחום ההצפנה המידע מוצפן והמערכת בודקת שהקובץ מגיע בשלמותו.
* המערכת לא מטפלת בתוכן הקובץ, ולא בודקת שלא נשלחים וירוסים או דברים זדוניים. המערכת גם מוגבלת לשרת הפנימית של בית הספר, ולא מאפשרת לקוחות חיצוניים מחוץ לרשת .

1. **קיימות**

הפרויקט מתקשר לקיימות כדור הארץ בכך שהופך את הקבצים הדיגיטליים ליותר נגישים ומהירים ובכך מעודד שימוש בקבצים דיגיטליים ובכך מוריד את השימוש בקבצים על הנייר וזה שומר על העצים ועל כדור הארץ. כמו כן הורדה מהירה יותר של הקבצים חוסכת חשמל, שימוש ברשת ובמחשבים.

## פרק ב' - nexus - אפיון

### פרוט המערכת:

* תיאור מפורט יותר של המערכת - נקסוס היא רשת של לקוחות שמוכנים ומעוניינים לתקשר אחד עם השני בשביל האופציה להוריד קבצים אחד מהשני בשביל מהירות הורדה גבוהה יותר. הרעיון להוריד חלק שונה של הקובץ המבוקש במקביל ממשתמשים שונים ולהרכיבו מחדש ככל שיהיו יותר משתמשים שהורידו את הקובץ מהירות ההורדה תלך ותגדל
* מה היא אמורה לעשות -בעת בקשת הורדת קובץ, המערכת תזהה את כל המשתמשים שכבר הורידו את הקובץ המבוקש אליהם והיא תוריד מכל אחד מהם חלק אחר – כך שתהליך ההורדה יבוצע במקביל, תבנה את הקובץ מחדש אצל המשתמש תוך הבטחת שלמות ותציג אותו למשתמש.

1. מה היכולות שהיא תעניק למשתמש , פירוט היכולות:

* הורדת קבצים – המערכת תאפשר ללקוח להוריד קבצים במהירות גבוהה יותר תוך מתן ערובה למהימנות ושלמות הקבצים המורדים.
* אבטחה – המערכת תאפשר לזהות קובץ פגום, כבר במהלך ההורדה, ולא רק בסיום ההורדה, ותתריע בהתאם למשתמש.

### פירוט הבדיקות ('קופסא שחורה')

(בדיקות לביצוע אך ורק עם קובץ ההרצה, ללא הקוד, בדגש על כך שכל אחד לפי הכתוב בטבלה יצליח לבצע את הבדיקה)

| **מספר** | **שם הבדיקה (שם שיעיד על תוכן הבדיקה)** | **מה אמורה לבדוק** | **איך מתכננים לבדוק**  **(לתאר בפירוט את שלבי הבדיקה)** |
| --- | --- | --- | --- |
| 1 | הורדת קובץ ממחשב אחד | שמחשב מצליח להוריד קובץ | לחבר 2 לקוחות שלאחד מהם אין קובץ מסוים לאחד מהם יש, ולנסות להוריד את הקובץ למחשב שאין לו את הקובץ בסיום ההורדה לפתוח את הקובץ ולבדוק שהוא זהה למקור. |
| 2 | הורדת קובץ מכמה מחשבים | שהמחשב מוריד את הקובץ במהירות גדולה יותר בעזרת כמה מחשבים | לחבר כמה מחשבים שלכולם יש את אותו הקובץ ואז לחבר מחשב נוסף שאין לו את הקובץ ולבקש להוריד את הקובץ  ולמדוד את הזמן שלקח לו להוריד את הקובץ, להשוואת את הזמן לזמן שלוקח ללקוח להוריד את אותו הקובץ בעזרת מחשב אחד |
| 3 | התרעת על קובץ לא שלם | שהמערכת מתריעה ללקוח שחלק מהקובץ או הקובץ כולו שהוא הוריד ממחשב מסוים לא תואם את הקובץ שהוא היה אמור להוריד | לחבר 2 מחשבים לרשת לבקש להוריד קובץ מסוים, המחשב שמורידים ממנו את הקובץ צריך לערוך את הקובץ שהוא לא יהיה שלם, ולבדוק האם המערכת מצליחה לזהות את הבעיה. |
| 4 | בדיקת פירוק ואיחוד קובץ | שיש הצלחה לפרק קובץ לחלק אותו לכמה מחשבים, ואז לקבל את החלקים מהמחשבים חזרה ולחבר | לחבר 2מחשבים ולהוריד את קובץ ולבדוק את הזמן שלוקח לו להוריד  לאחר מכן לפתוח מחשב נוסף להוסיף לו את הקובץ, ולמחוק את הקובץ שהורדת בפעם הראשונה אצל המחשב הראשון ואז לבקש שוב להוריד את אותו הקובץ ולבדוק האם לוקח לו פחות זמן העובדה שזמן ההורדה קטן מעידה על זה שהקובץ ירד מכמה מחשבים ושהפירוק והאיחוד עובד |
| 5 | בדיקה של קישור ממשק לקוח למערכת | שאחרי שהלקוח מבקש קובץ מסוים ההודעה עוברת בתקשורת והוא מקבל את הקובץ | לפתוח לקוח ולבדוק שמופיע הממשק משתמש ולבקש ממנו להוריד קובץ ולבדוק האם הקובץ יורד |
| 6 | עבודה במקביל | שניתן להוריד כמה קבצים במקבלים (כמה משתמשים שונים ) | לפתוח 2 מחשבים ולבדוק כמה זמן לוקח להם להוריד קובץ מסוים, לאחר מכן למחוק את הקובץ ממחשב אחד, ואז לפתוח כמה מחשבים ולבקש להוריד את אותו הקובץ מהפעם הראשונה ובנוסף לזה לבקש קובץ אחר ממחשב אחר ולבדוק שהעבודה במקביל לא משפיעה על הזמן של הורדת הקובץ הראשון |
| 7 | הורדה במקבילים | שניתן להוריד את אותו הקובץ במקביל אצל שני לקוחות או יותר | לפתוח כמה מחשבים ולבקש מ2 מהם קובץ שקיים בכמה מחשבים אחרים |
| 8 | מערכת עדכנית | שלא ניתן להוריד קובץ מהמערכת אם הוא נמחק מהלקוחות שהורידו אותו | לפתוח מחשב ולהוסיף אליו קובץ שקיים רק אצלו ואז לבדוק האם אפשר להוריד את הקובץ לאחר מכן למחוק את הקובץ בכל המחשבים שהקובץ קיים עליהם ולבדוק האם הוא עדיין מופיע במחשב שלישי שלא היה עליו את הקובץ |

### תכנון לוח זמנים לפרויקט

* תכנון לוח זמנים ראשוני
* עבור כל פעילות בלוח: זמן התחלה מתוכנן, זמן סיום מתוכנן , זמן התחלה בפועל, זמן סיום בפועל והערות. יש לכלול את התאריכים שניתנו לכם עבור כל פרק כ'זמן סיום מתוכנן' ואת החודש של הקידוד לפרק לאבנים גדולות ולתת להן זמנים

| **פעילות** | **זמן התחלה מתוכנן** | **זמן סיום מתוכנן** | **זמן התחלה בפועל** | **זמן סיום בפועל** | **הערות** |
| --- | --- | --- | --- | --- | --- |
| יזום | 6.11.2023 | 25.11.2023 | 6.11.2023 | 7.11.2023 |  |
| אפיון | 8.11.2023 | 1.12.2023 | 9.11.2023 | 12.11.2023 |  |
| ניתוח | 11.11.2023 | 15.12.2023 | 13.11.2023 | 23.11.2023 |  |
| עיצוב | 1.12.2023 | 15.1.2023 | 24.11.2023 | 14.1.2024 |  |
| קוד  1) חלוקת קובץ לחלקים במקביל שמירת hash  2) בניית קובץ מפורק במקביל בדיקת hash  3)יצירת קובץ torrent  4)מימוש תקשורת והעברת חלקי הקובץ  5)בניית ממשק משתמש  6)ניטור תיקיות  7)מימוש הודעה דרך ממשק המשתמש ובקשה דרך הממשק | 1)  15.1.2024  2)  15.1.2024  3)  22.1.2024  4)  1.2.2024  5)  14.2.2024  (6  21.2.2024  7)  28.2.2024 | 1)  22.1.2024  2)  22.1.2024  3)  1.2.2024  4)  14.2.2024  5)  21.2.2024  6)  28.2.2024  7)  1.3.2024 | 1)15.1.2024  2)19.1.2024  3)26.1.2024  4)29.1.2024  5)10.2.2024  6)16.2.2024  7)22.2.2024 | 1)18.1.2024  2)26.1.2024  3)29.1.2024  4)10.2.2024  5)31.2.2024  6)22.2.2024  7)31.2.2024 | כל אחד מהחלקים היה מוכן ע"פ תאריך הסיום אך היו שינוי גם לאחר מכן |
| גרסה ראשונית | 15.1.2024 | 1.3.2024 | 15.1.2024 | 1.3.2024 |  |
| מסמך בדיקות | 1.4.2024 | 10.4.2024 | 1.4.2024 | 11.4.2024 |  |
| מדריך למשתמש | 10.4.2024 | 1.5.2024 | 13.4.2024 | 16.4.2024 |  |
| גרסה סופית | 1.5.2024 | 10.5.2024 | 16.4.2024 | 28.4.2024 |  |
| סגירת תיק פרויקט | 1.5.2024 | 10.5.2024 | 7.5.2024 | 12.5.2024 |  |
| מצגת הסבר | 1.5.2024 | 10.5.2024 |  |  |  |

### ניהול סיכונים בפרויקט

* זיהוי סיכוני הפרויקט, ניתוחם ומענה עליהם, הטבלה תמוין מסיכון גבוה לנמוך (הגבוהים באדום, בינוני –כתום , קל – צהוב) סיכון שיוסר יצבע בירוק ויעבור לתחתית המסמך

| **הסיכון** | **פירוט הסיכון** | **רמת הסיכון**  **(קל/בינוני/קשה)** | **תיאור דרכים (לפחות 2 ) להתמודדות עם הסיכון ולהקטין אותו** | **מה בוצע בפועל** | **תאריך** |
| --- | --- | --- | --- | --- | --- |
| אי עמידה בזמנים | פרויקט לא יושלם | קשה | * הקדמת לו"זים משימות * ארגון הזמן בצורה יעילה * להתחיל בדברים היותר קשים | הפרויקט הושלם | 12.5 |
| אי הצלחה להרכיב קובץ חזרה מחלקים | הפרויקט לא יושלם | קשה | * לבדוק שאפשר לממש את זה * לקרוא באינטרנט על פתרונות של אנשים ועל דרכי הפעולה שלהם | הצלחה להרכיב את הקובץ | 26.1.2024 |
| אי הצלחת פירוק הקובץ | פרויקט לא יהיה שימושי ובלי תועלת | קשה | * לבדוק שאפשר לממש את זה * לקרוא באינטרנט על פתרונות של אנשים ועל דרכי הפעולה שלהם | הצלחה לפרק את הקובץ | 18.1.2024 |
| הקובץ ירד לאט | הפרויקט יהיה פחות יעיל | בינוני | כתיבת קוד יעיל מההתחלה  כתיבת אלגוריתם חכם | הקוד נכתב בצורה יעילה |  |
| לא להצליח להבין את הניהול הגרפי | הפרויקט לא יראה טוב ויהיה הרבה פחות מזמין | קל | לקרוא באינטרנט לנסות להגיע להבנה עמוקה של המחלקה  להתייעץ עם חברים שהצליחו והגיעו לתוצרים יפים | יצירת הגרפיקה הייתה עם השקעה רבה והתוצר נראה מעולה |  |

## 

## פרק ג' - Nexus- מסמך ניתוח

### פרוט יכולות המערכת

1. לעלות קובץ חדש למערכת - קובץ זה יכול להיות קובץ שקיים במערכת או קובץ חדש שהלקוח יצר והוא מעוניין לשתף אותו עם המערכת
2. להוריד קובץ שקיים במערכת - הלקוח מקבל רשימה עם קבצים שקיימים במערכת והוא יכול לבחור איזה אחד מהם הוא רוצה להוריד, אותו הקובץ ירד במהירות גבוהה.
3. פירוק קובץ – מפרקים את הקובץ לחלקים שווים ושמירת הhash של כל קובץ ושל הקובץ הכללי
4. חיבור קובץ - לאחר קבלת כל החלקים של הקובץ לאחר בדיקת כל חלק קובץ עם הhash הקיים ובדיקה הקובץ הסופי עם הhash הסופי
5. בניית קובץ torrent - בניית קובץ torrent עם ה headerהמתאימים לאחר השגתם מפירוק הקובץ
6. עדכון קובץ torrent - עדכון header בתוך קובץ הtorrent לאחר שלקוח שינה את הסטטוס של שמירת הקובץ אצלו (מחק או הוסיף את הקובץ) או לחילופין משתתפי הקובץ לא מחוברים
7. יצירת רשימה של קבצי המערכת – הרשימה נמצאת אצל השרת לאחר שקיבל את הנתונים מכל הלקוחות, כל שינוי ברשימה ישלח לכל לקוח.
8. שליחת חלקי הקובץ – כל לקוח הוא שרת שיכול לקבל בקשה מלקוח אחר להוריד חלק מקובץ שקיים אצלו, לאחר הבקשה הלקוח ישלח את חלק הקובץ לאותו לקוח
9. קבלת חלקי הקובץ – כל לקוח יכול לקבל מהשרת אצל לקוח אחר חלק מקובץ, לאחר הקבלה הלקוח יוודא שהחלק תקין על ידי hash
10. ניטור תיקייה – כל לקוח מנטר תיקייה מוסתרת (שהיא תיקיית המערכת שבה נשמרים הקבצים המורדים), את תוצאות הניטור הלקוח שולח לשרת.
11. ממשק משתמש – מציג רשימת קבצים קיימים במערכת, מאפשר בחירת קובץ מהרשימה הזו להורדה ומאפשר העלאת כל קובץ מכל מקום במחשב המשתמש.  
    כמו כן הממשק יציג חיווים על הצלחת/כשלון העלאת קבצים ועל הצלחה/כשלון הורדת קבצים למשתמש.
12. התחברות לשרת –  במסגרת פעולת ההתחברות, הלקוח ישלח לשרת את רשימת הקבצים הקיימים אצלו בתיקייה המנוטרת ובמקביל יקבל מהשרת את רשימת הקבצים הזמינים במערכת.
13. תקשורת Peer2Peer – מול כל אחד מאלו שהקובץ המבוקש זמין אצלם

שם היכולת: פירוק קובץ

מהות היכולת: פירוק קובץ לחלקים שווים

אוסף יכולות:

* חישוב hash לכל הקובץ
* פירוק הקובץ לחלקים שווים
* מספור כל חלק קובץ
* חישוב hash לכל חלק
* הצפנת כל חלק

רשימת אובייקטים: הצפנה/פענוח , קובץ , hash

שם היכולת: חיבור קובץ

מהות היכולת: חיבור חלקי קובץ לקובץ שלם

אוסף יכולות:

* פיענוח כל חלק
* השוואת hash של כל חלק
* חיבור החלקים לפי מספור
* השוואת hash של כל הקובץ

רשימת אובייקטים: הצפנה/פענוח, torrent file, קובץ, hash

שם היכולת: העלאת קובץ למערכת

מהות היכולת: הוספת קובץ שניתן להורדה למערכת

אוסף יכולות:

* **ממשק משתמש** – בחירת הקובץ להעלאה
* בניית הודעה על פי פרוטוקול
* הצפנה של הקובץ
* שליחת הקובץ לגורם מרכזי
* קבלת אישור שהעלאה הצליחה
* ברקע – לאחר קבלת האישור מהשרת העתקת הקובץ הנבחר לתיקיית הפרויקט (נסתרת) המנוטרת

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת, פרוטוקול, קובץ, תיקייה מנוטרת

שם היכולת: בניית קובץ torrent

מהות היכולת: בניית קובץ Torrent המכיל פרטים רלוונטיים

אוסף יכולות:

* **פירוק הקובץ**
* בניית קובץ ה Torrent עם כל המידע הנחוץ (שם קובץ, כמות חלקים, Hash של כל חלק, hash של הקובץ כולו, רשימת כתובות מהם ניתן להוריד את הקובץ)
* עדכון בסיס נתונים עם שם הקובץ
* שמירת קובץ ה Torrent אצל הגורם המרכז

רשימת אובייקטים: הצפנה/פיענוח, תבנית קובץ torrent, בסיס נתונים, תיקיית קבצי ה Torrent

שם היכולת: עדכון קובץ torrent

מהות היכולת: עדכון פרטים בתוך קובץ הtorrent

אוסף יכולות:

* פתיחת הקובץ
* עדכון רשימת משתתפים

רשימת אובייקטים: קובץ torrent רשימת משתתפים

שם היכולת: יצירת רשימת קבצים להורדה

מהות היכולת: יצרת רשימת קבצים ניתנים להורדה מהמערכת

אוסף יכולות:

* קבלת מכל לקוח ברשת את רשימת הקבצים הקיימים אצלו בתיקייה המנוטרת
* אימות הנתונים מול בסיס הנתונים:
  + שם קובץ שלא קיים בבסיס נתונים משמעותו שלא קיים עבורו קובץ Torrent - נשלחת למחשב הספציפי הודעה למחוק את הקובץ מהתיקייה המנוטרת
  + שם קובץ שקיים שקיים בבסיס הנתונים משמעותו שקיים עבורו קובץ Torrent– הוספת ה IP לקובץ הtorrent (במידה ולא קיים)
* במידה ושם הקובץ נוסף לראשונה והיה קיים קובץ torrent אז לעדכן את הרשימה

רשימת אובייקטים: תיקייה מנוטרת, רשימת קבצים, תקשורת, בסיס נתונים , קובץ Torrent

שם היכולת: הורדת קובץ מהמערכת

מהות היכולת: הורדה מהירה של קובץ מהלקוחות האחרים במערכת

אוסף יכולות:

* **ממשק משתמש** – בחירת הקובץ להוריד
* קליטת נתונים
* בדיקת תקינות
* בניית הודעה על פי פרוטוקול
* הצפנה
* שליחה לשרת
* קבלת תשובה קובץ torrent מהשרת
* בקשת **שליחת כל חלקי הקובץ**
* קבלת תשובה
* **חיבור קובץ**
* שמירת הקובץ בdownload והעתקתו לתיקייה המנוטרת

רשימת אובייקטים: ממשק משתמש, הצפנה/פיענוח, תקשורת, torrent file , פרוטוקול

שם היכולת: שליחת חלקי קובץ

מהות היכולת: שליחת חלק ספציפי מהקובץ

אוסף יכולות:

* קבלת בקשה
* פענוח
* פירוק לפי פרוטוקול
* פתיחת הקובץ המבוקש
* מציאת החלק המתבקש
* בניית תשובה לפי פרוטוקול
* הצפנה
* שליחת חלק הקובץ

רשימת אובייקטים: תקשורת, קבצים, הצפנה, פרוטוקול

שם היכולת: ניטור תיקייה

מהות היכולת: ניטור התיקייה ושליחת עדכונים לשרת

אוסף יכולות:

* יצירת התיקייה במידה ואין
* בדיקת שינויים
* בניית הודעה על פי פרוטוקול
* הצפנת השינויים
* שליחת השינויים לשרת
* קבלת הנחיות מהשרת

מחיקת קובץ – מוחקים את הקובץ הספציפי מהתיקייה המנוטרת

הוספת קובץ מעתיקים את הקובץ הספציפי לתיקייה המנוטרת

* רשימת אובייקטים: תקשורת, קבצים, הצפנה , פרוטוקול תיקייה מנוטרת

שם היכולת: הצגת רשימת הקבצים

מהות היכולת: הצגת ממשק המערכת למשתמש – העלאת הורדת קובץ

אוסף יכולות:

* קבלת הרשימה של הקבצים מהשרת בכל רגע נתון והצגתם למשתמש
* במידה ונבחר העלאת קובץ - הצגת הקבצים במחשב ואפשרות בחירה למשתמש - **העלאת קובץ**
* במידה ונבחר הורדת קובץ – **הורדת קובץ מהמערכת**
* רשימת אובייקטים: תקשורת, קבצים, הצפנה, גרפיקה

## פרק ד' - Nexus - העיצוב

### תיאור הארכיטקטורה של המערכת המוצעת

* תיאור החומרה –

שרת: מחשב מבוסס SQL + Windows בסיס נתונים

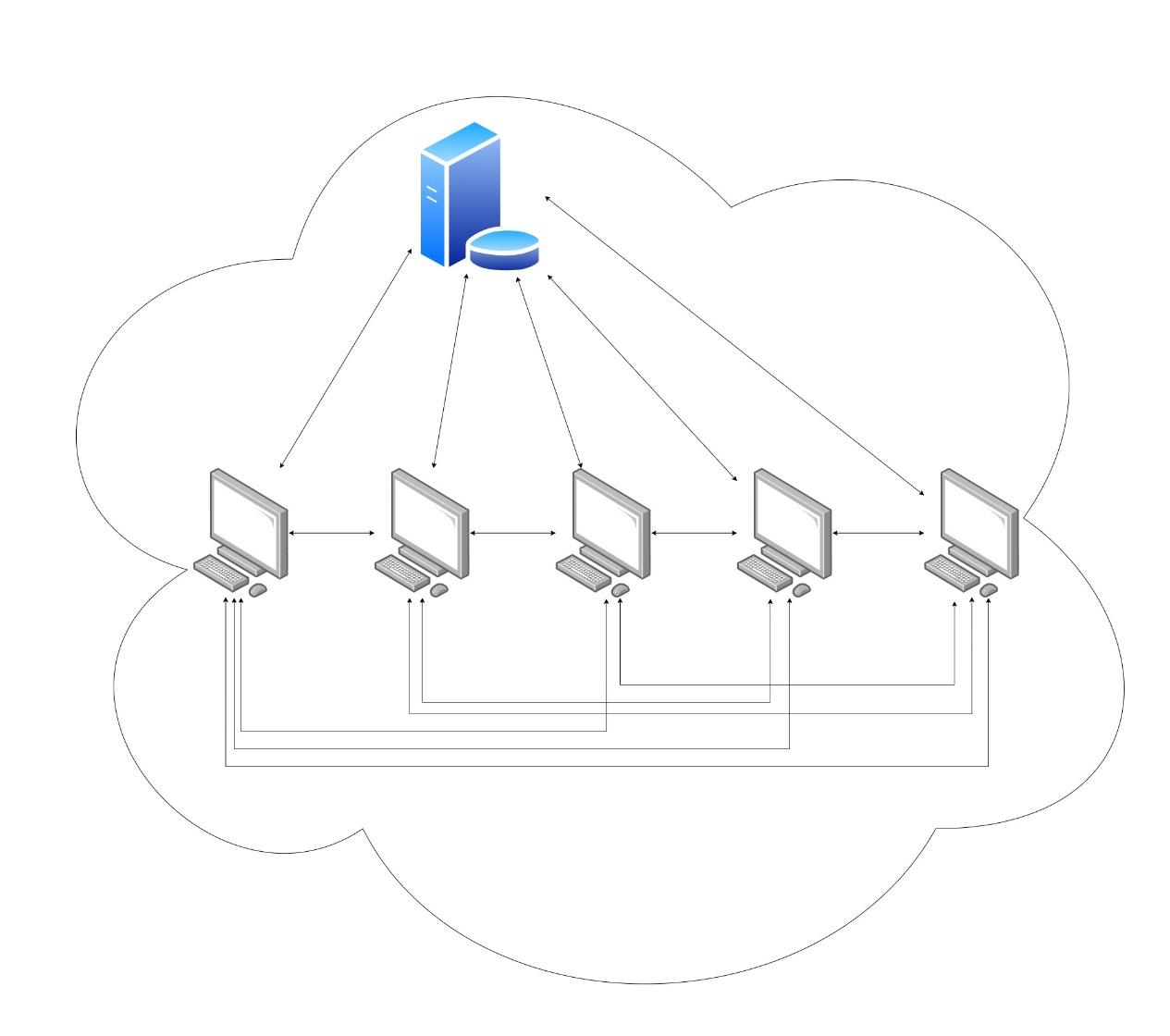
לקוחות: מחשבים מבוססים Windows

כל המחשבים (שרת ולקוחות) נמצאים על אותה הרשת

* רכיבים שונים והקשרים ביניהם

כל הלקוחות מתחברים לשרת(שרת מרובה לקוחות) , שליחת הקבצים בעזרת P2P

לכל לקוח יש רכיב של שרת מרובה לקוחות



### תיאור הטכנולוגיה הרלוונטית

* פירוט ההחלטות שנלקחו בחשבון בעת בחירת הטכנולוגיה

**שפת תכנות** – Python 3.7, DB SQL

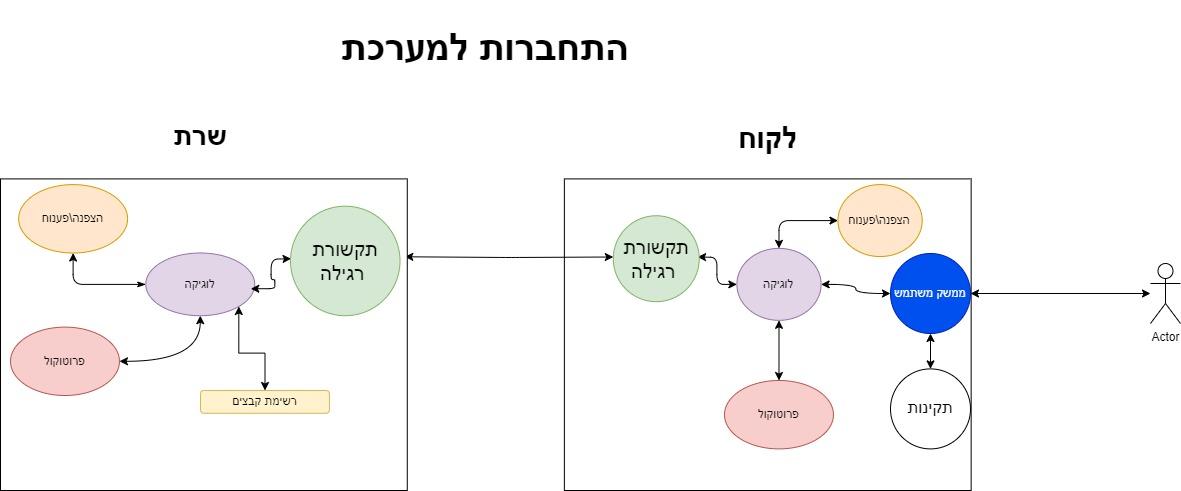
**מערכת הפעלה** - Windows

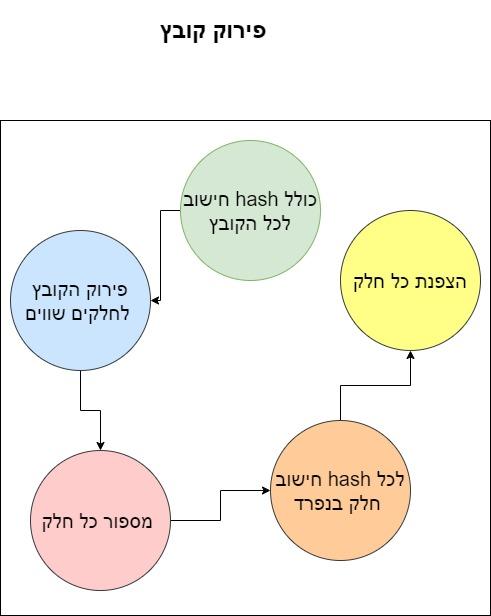
**תקשורת** – תקשורת מבוססת Sockets בעזרת פרוטוקול TCP

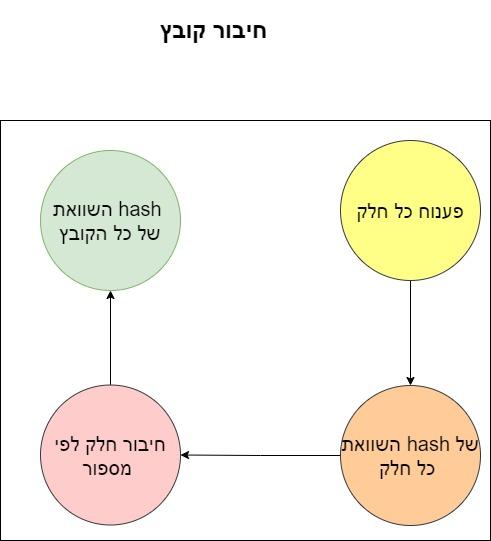
**תחומי עניין** – הצפנה (כל מה שעובר ברשת מוצפן) , בסיס נתונים טבלאי( SQL)

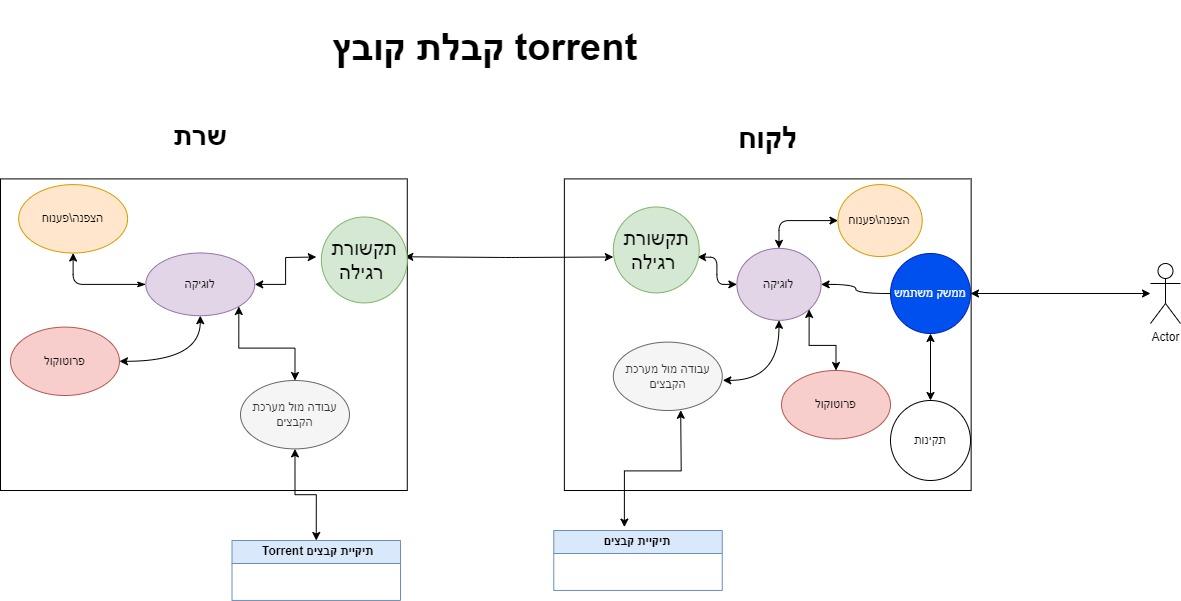
גרפיקה עיצוב בעזרת WXPython , עבודה עם קבצים, עבודה עם hash

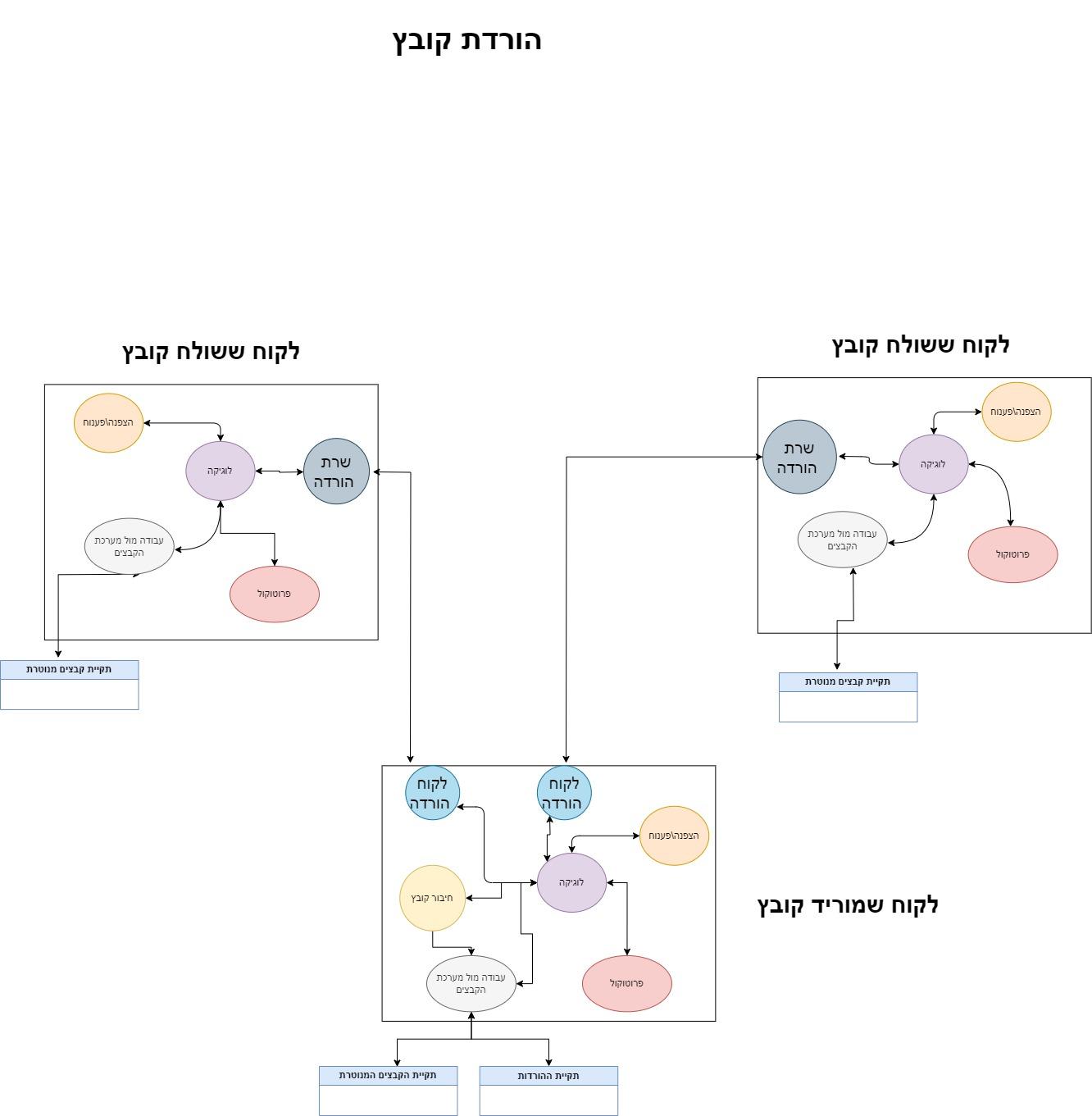
1. תיאור מודולים בהם נעשה שימוש

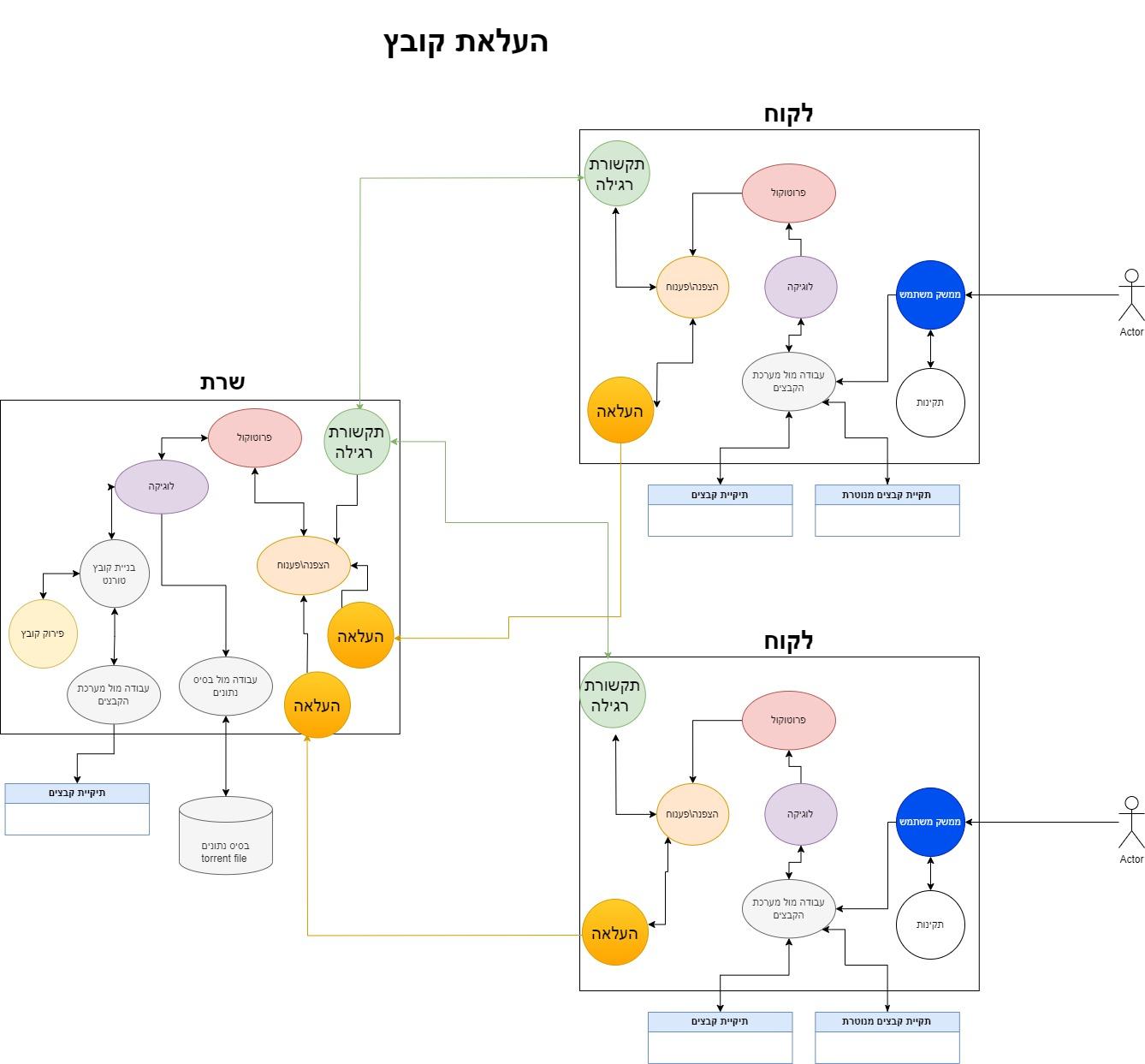


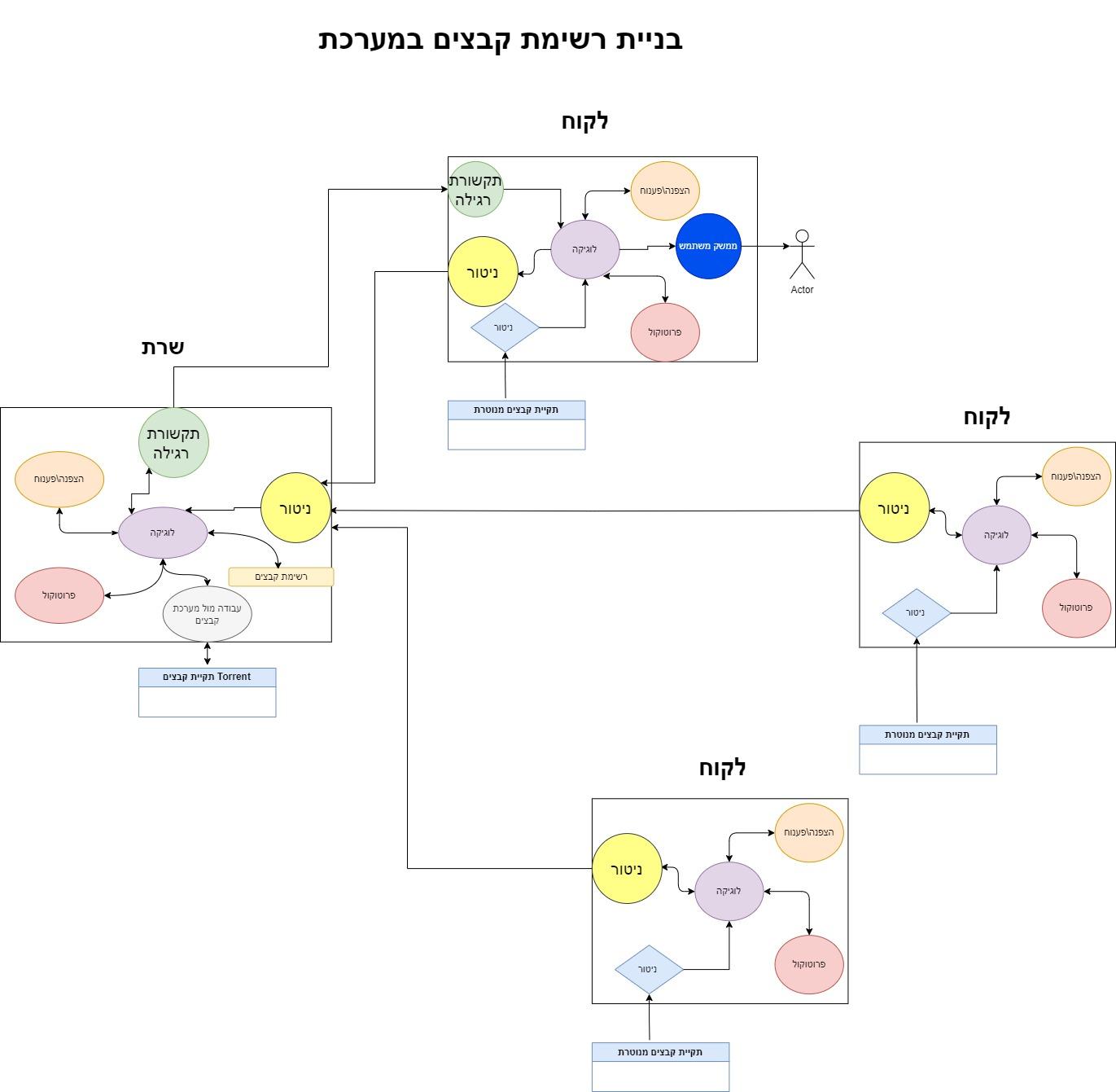


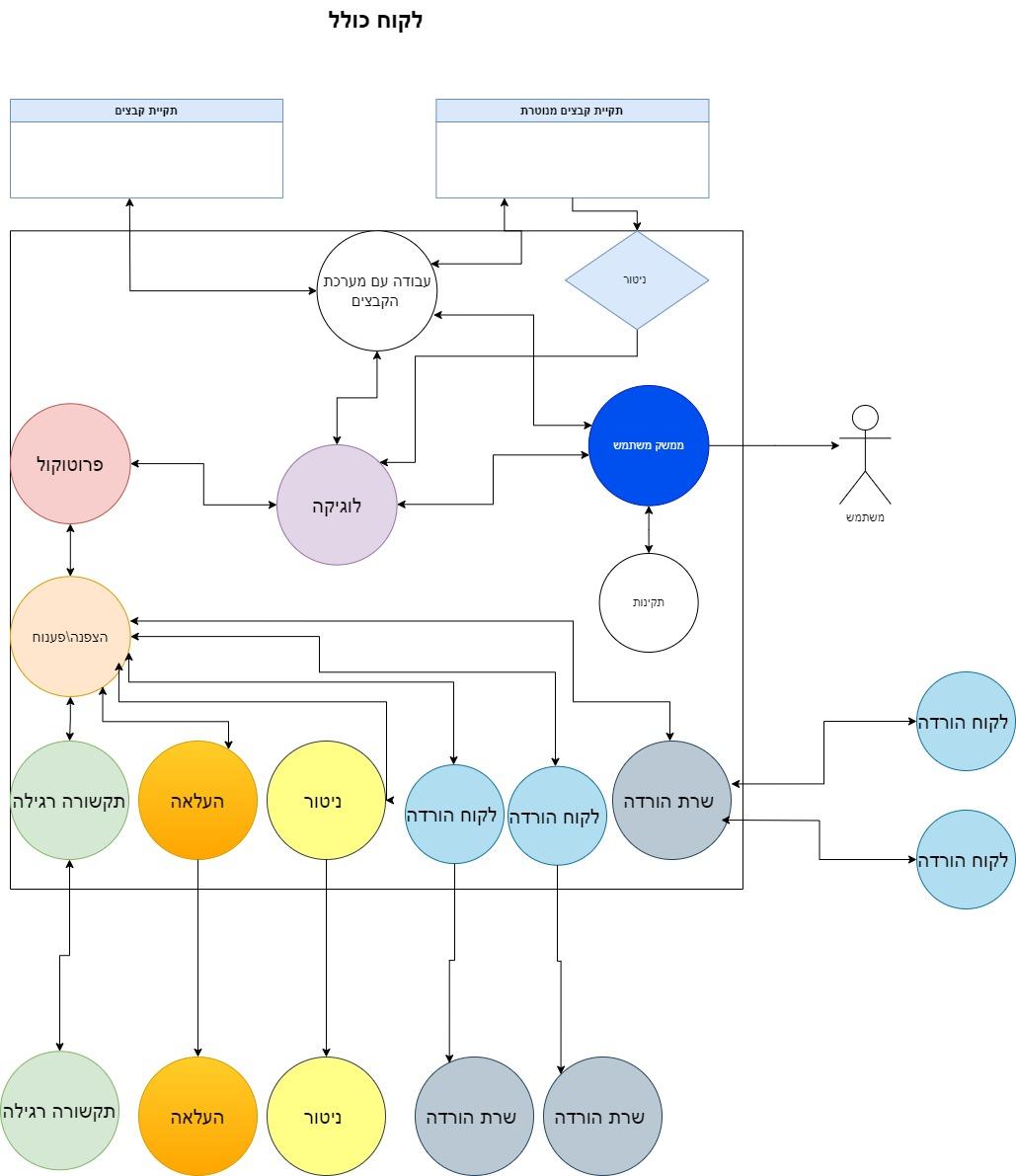


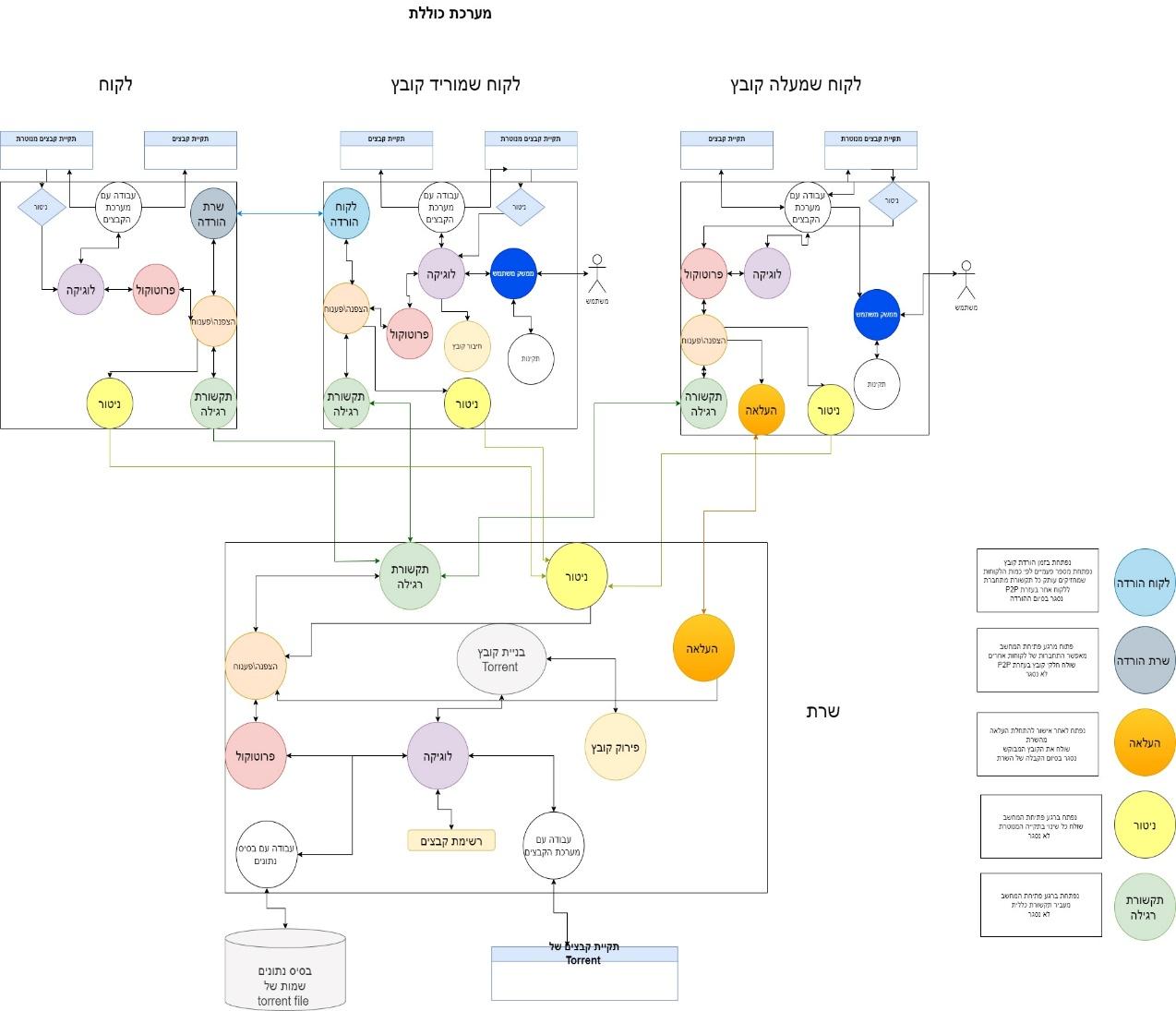












* סקירת כל המודולים המרכיבים את המערכת וקשרי הגומלין ביניהם:

**מודולים מיובאים לפרויקט:**

| **חבילה** | **ייעוד** |
| --- | --- |
| Socket | תקשורת |
| Hashlib | הצפנה |
| Select | שרת מרובה לקוחות |
| Wxpython | Gui |
| Base64 | הצפנה |
| AES | הצפנה |
| random | הצפנה |
| Threading | ריצה במקביל |
| sys | שימוש בפעולות של המפרש (interpreter) |
| os | שימוש בפעולות של מערכת ההפעלה |
| humanize | גרפיקה להפוך מספר לאנושי |
| queue | תקשורת בין Threading |
| Json | שמירת קבצי הtorrent |
| pubsub | תקשורת עם הגרפיקה |
| multiprocessing | ריצה על כמה ליבות |
| Win32file | Win api monitoring |
| Win32con | Win api monitoring |
| Win32event | Win api monitoring |
| ctypes | Win api monitoring |
| Sqlite3 | חבילת DB |
| math | פעולות מתמטיות |

**מודולים שלי בפרויקט:**

### מודולים בשרת

| **serverProtocol** | | |
| --- | --- | --- |
| מחלקה מאגדת את כל הפעולות הנדרשות לטיפול בפרוטוקול בצד שרת | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
|  |  | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| unpack | data – בתים שהתקבלו בתקשורת | מפרקת את ההודעה ע"פ הפרוטוקול ומחזירה tuple:  opcode – קוד ההודעה  List\_par – רשימת פרמטרים |
| Unpack\_file | data – בתים שהתקבלו בתקשורת | מפרקת את ההודעה ע"פ הפרוטוקול ומחזירה tuple:  opcode – קוד ההודעה  List\_par – רשימת פרמטרים |
| build\_part\_of\_key | b – מספר ראשוני | בונה הודעה העברת מפתח ע"פ הפורמט ומחזיר אותה  00b |
| Response\_for\_torrent\_request | File\_name – שם הקובץ | בונה הודעה העברת קובץ טורנט ע"פ הפורמט ומחזיר אותה |
| Response\_for\_upload\_request | Path\_of\_file– שם הקובץ  Port – הפורט של הסוקט הנפתח | בונה הודעה פתיחת סוקט העלאה ע"פ הפורמט ומחזיר אותה |
| Response\_for\_upload | - Bolהאם העלאה עבדה  Path – הדרך לקובץ | בונה הודעה אישור קבלת הקובץ ע"פ הפורמט ומחזיר אותה |
| Delete\_file\_from\_folder | File\_name – שם הקובץ | בונה הודעה הוראת מחיקת קובץ ע"פ הפורמט ומחזיר אותה03file\_name |
| Create\_string\_of\_list | list\_of\_open\_files | בונה הודעה ע"פ פרוטוקול ומחזיר אותה |

| **mainServer** | | |
| --- | --- | --- |
| מחלקה שמאגדת את פעולות למימוש הלוגיקה בשרת הראשי | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| General\_comm | התקשרות הכללית | |
| Nitur\_comm | עצם תקשורת של שרת הניטור | |
| general\_queue | תור הודעות שהגיעו לשרת הכללי | |
| nitur\_queue | תור הודעות שהגיעו לשרת הניטור | |
| general\_commands | מילון פעולות כלליות (עבור השרת הכללי) בפורמט general\_commands["opcode"]=proc\_name | |
| nitur\_commands | מילון פעולות כלליות (עבור שרת הניטור) בפורמט nitur\_commands["opcode"]=proc\_name | |
| list\_of\_open\_files | רשימה ששומרת את כל הקבצים שניתן להוריד אותם | |
| unused\_ports | שומר את כל הפורטים בתוך גנרטור | |
| Files\_obj | אוביקט של מחלקת הקבצים בשרת | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| main |  | פעולה ראשית:  יוצרת תורים עבור ההודעות שמגיעות מהשרת הכללי ומשרת הניטור  יוצרת את השרת הראשי  יוצרת את שרת הניטור  מפעילה כthread את handle\_general\_msgs  מפעילה כ thread את handle\_nitur\_msgs |
| handle\_general\_msgs  (מופעל כ thread ) | general\_queue | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  מפעילה פעולה רלוונטית ע"פ general\_commands["opcode"]=proc\_name |
| handle\_nitur\_msgs  (מופעל כ thread ) | nitur\_queue | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  מפעילה פעולה רלוונטית ע"פ nitur\_commands["opcode"]=proc\_name |
| handle\_upload\_file  (מופעל כ thread ) | File\_name  Server\_comm\_upload  sever  queue  Ip | יוצרת עצם בסיס הנתונים db  מוציא את הקובץ מהqueue  מפעיל את הפעולה serverfile.Create\_torrent\_file  נקראת הפעולה db.add\_torrent  נקראת הפעולה serverprotocol. Response\_for\_upload  נשלח הודעת אישור ללקוח בעזרת server.send  ונסגר העצם של ה Server\_comm\_upload |
| Delete\_file\_list\_of\_open\_files | Name\_of\_file  server  ip | מוחק את השם של הקובץ מרשימה  קורא לפעולה serverfile. delete\_ip\_from\_torrent  במידה וחזר true  יוצר הודעה בעזרת serverProtocol. Create\_string\_of\_list  ושולח לכולם את העדכון בעזרת server.sendall |
| File\_added | Params – רשימה של משתנים שהפעולה משתמשת בהם | אם לקובץ אין טורנט נשלחת הודעת שגיאה לשרת  אחרת  מוסיף את השם של הקובץ לרשימה  קורא לפעולה serverfile.  Add\_ip\_to\_torrent  יוצר הודעה בעזרת serverProtocol. Create\_string\_of\_list  ושולח לכולם את העדכון בעזרת server.sendall |
| Delete\_file\_in\_client | Params – רשימה של משתנים שהפעולה משתמשת בהם | קורא לפעולה serverfile. delete\_ip\_from\_torrent  קורא לפעולה serverprotocol. Delete\_file\_from\_folder  שולח הודעה ללקוח שימחק את הקובץ מהתיקייה המנוטרת בעזרת server.send |
| File\_changed | Params – רשימה של משתנים שהפעולה משתמשת בהם | אם יש טורנט מוריד את הIP מהרשימה של הלקוחות הפתוחיםת מעדכן את הרשימה ושולח עדכון.  שולח לאותו לקוח למחוק את הקובץ |
| File\_name\_changed | Params – רשימה של משתנים שהפעולה משתמשת בהם | שולח לאותו לקוח למחוק את הקובץ |
| Send\_list\_of\_file | Params – רשימה של משתנים שהפעולה משתמשת בהם | שולח את רשימת הקבצים ללקוח |
| send\_torrent | Params – רשימה של משתנים שהפעולה משתמשת בהם | משתמש ב server\_files. Get\_torrent\_file(filename)  קורא לפרוטוקול Response\_for\_torrent\_request  ואז שולח server.send |
| Create\_upload\_socket | Params – רשימה של משתנים שהפעולה משתמשת בהם | יוצרת עצם בסיס הנתונים db  יוצר queue מקבל פורט מתוך הגרנטור unused\_port יוצר עצם מתוך servercomm  ןמפעיל כ process את הפעולה  handle\_upload\_file(queue, Server\_comm\_upload)  יוצר הודעה בעזרת  Response\_for\_upload\_request  ושולח הודעה לip שביקש בעזרת server.send |
| \_after\_upload | List\_queue | שולח תשובה ללקוח לאחר העלאת קובץ |
| Handle\_upload\_file | Upload\_queue – התור שמחובר לרשת  File\_obj – עצם הקבצים  Port – הפורט  Update\_list\_queue­­ |  |
| Handle\_disconnect\_nitur | Params - רשימה של משתנים שהפעולה משתמשת בהם | מוחק את הip של הלקוח מכל הtorrent files, מעדכן את הרשימה של הלקוחות ושולח לכולם |
| Create\_open\_files |  | יוצר רשימה של כל הקבצים שנמצאים בתיקיית קבצי הטורנט |

| **ServerFiles** | | |
| --- | --- | --- |
| מחלקה מאגדת את כל הפעולות שנדרשות לטיפול במערכת הקבצים בשרת | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
|  |  | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| Create\_torrent\_file | File\_name  File\_data | מחושב hash לכל הקובץ ולכל חלק  קורא לפעולה \_Build\_torrent\_file |
| Build\_torrent\_file\_ | File\_data – הdata של קובץ הtorrent  File\_name  Full\_hash  Parts\_hash  Len\_of\_part | יוצר קובץ torrent כפורמט של קובץ json עם הפרמטרים הנחוצים |
| Add\_ip\_to\_torrent | File\_name – שם הקובץ  Ip - הכתובת | במוסיף לשדה של הip הפתוחים בתוך קובץ הtorrent את הקובץ |
| Get\_torrent\_file | file\_name – שם הקובץ  String – האם להחזיר את הקובץ כמחרוזת | מחזיר מחרוזת של הקובץ torrent בפורמט json ע"פ פרוטוקול הקובץ ואם הקובץ לא קיים מחזיר None |
| delete\_ip\_from\_torrent | File\_name – שם הקובץ  Ip - הכתובת | מוריד מהשדה של הip הפתוחים בתוך קובץ הtorrent את הקובץ במידה בקיים  מחזיר Boolean אם הקובץ קיים |
| Handle\_disconnect | ip | מוריד את הip מכל קבצי הטורנט שהip קיים בהם |
| Delete\_temp | File\_name | מוחק את הקובץ הטורנט |

|  |  |  |
| --- | --- | --- |
| **DB** | | |
| מחלקה היוצרת עצם לטיפול וניהול בסיס הנתונים של הפרויקט | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| dbName | שם בסיס הנתונים " NexusDB" | |
| tblName | שם הטבלה " torrentTBL " | |
| Conn | מצביע לבסיס הנתונים שנוצר | |
| Curr | מצביע לטבלה | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ |  | מאתחלת את המשתנים ומפעילה את \_buildDB |
| \_buildDB |  | מתחברת לבסיס הנתונים ויוצרת (אם לא קיימת) את הטבלה עם השדות היעודיים |
| add\_torrent | Torrent\_file\_name – שם הקובץ | מוסיף את שם הקובץ לבסיס הנתונים במידה ולא קיים (שימוש ב have\_torrent) במידה וקיים מחזיר None |
| have\_torrent | Torrent\_file\_name – שם הקובץ | מחזיר Boolean האם השם קיים בבסיס הנתונים |
| DeleteDB |  | מוחק את הdb |
| closeDb |  | סוגר את הdb |

### מודולים בלקוח

| **ClientFiles** | | |
| --- | --- | --- |
| מחלקה המאגדת את כל הפעולות שנדרשות לטיפול במערכת הקבצים בלקוח | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
|  |  | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| Get\_part\_of\_file | Path of file – הדרך לקובץ  Number of file - מספר חלק | Data – החלק  במקרה שהחלק הוא מספר 1- מוחזר כל הData |
| save\_file | Path of file – הדרך לקובץ  Name of file - שם קובץ  Data – המידע של הקובץ | שומר את הקובץ |
| Delete\_file | Path of file – הדרך לקובץ | מוחק את הקובץ |

| **ClientProtocol** | | |
| --- | --- | --- |
| מחלקה מאגדת את כל הפעולות הנדרשות לטיפול בפרוטוקול בצד הלקוח | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
|  |  | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| unpack | data – בתים שהתקבלו בתקשורת | מפרקת את ההודעה ע"פ הפרוטוקול ומחזירה tuple:  opcode – קוד ההודעה  List\_par – רשימת פרמטרים |
| Unpack\_file | Data – בתים שהתקבלו בתקשורת | מפרקת את הודעת הקובץ ע"פ פרוטוקול ומחזירה tuple:  opcode – קוד ההודעה  List\_par – רשימת פרמטרים |
| build\_part\_of\_key | A– מספר ראשוני | בונה הודעה העברת מפתח ע"פ הפורמט ומחזיר אותה 00A |
| Request\_torrent\_file | File name – שם הקובץ | בונה הודעת בקשה של קובץ טורנט ע"פ פרוטוקול |
| Request\_upload | path– הדרך לקובץ | בונה הודעת בקשת העלאה של קובץ חדש למערכת ע"פ פרוטוקול |
| Upload\_file | File name – שם הקובץ | בונה הודעת העלאת קובץ חדש למערכת ע"פ פרוטוקול |
| Request\_part\_file | File name – שם הקובץ  Number\_of\_part – מספר הקובץ  Len\_of\_part – גודל החלק | בונה הודעת בקשת חלק קובץ מלקוח אחר ע"פ פרוטוקול |
| Send\_file\_part | Number\_of\_part – מספר הקובץ  Data – המידע של הקובץ | בונה הודעת שליחת חלק קובץ ללקוח שביקש ע"פ פרוטוקול |
| Added\_file\_nitur | File\_name – שם הקובץ | בונה הודעה שנוסף קובץ לתיקיית הניטור |
| Removed\_file\_nitur | File\_name – שם הקובץ | בונה הודעה שנמחק קובץ בתוך תיקיית הניטור |
| Changed\_file\_nitur | File\_name – שם הקובץ | בונה הודעה שהשתנה קובץ בתוך תיקיית הניטור |
| Changed\_file\_name\_new\_nitur | File\_name – שם הקובץ | בונה הודעה עם השם החדש של הקובץ |

­

| **mainClientUser** | | |
| --- | --- | --- |
| מחלקה שמאגדת את פעולות למימוש הלוגיקה בשרת הראשי | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| gui\_queue | התור שאליו נכנסות הודעות מהמשתמש | |
| general\_queue | תור הודעות שהגיעו לשרת הכללי | |
| General\_commands | מילון ששומר את הפעולות ע"פ opcode שמגיע מהשרת | |
| Server\_ip | Ip של השרת | |
| abort | בוליאן האם להספיק את ההורדה של קובץ | |
| frame | עצם של הגרפיקה | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| main |  | פעולה ראשית:  יוצר את הגרפיקה  יוצרת תורים עבור ההודעות שמגיעות מהשרת הכללי ומהגרפיקה  ומפעיל כthread את הפעולות :  Get\_message\_from\_gui  handle\_general\_msgs |
| handle\_general\_msgs  (מופעל כ thread ) | general\_queue –ההודעות מהשרת | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  מפעילה פעולה רלוונטית ע"פ general\_commands["opcode"]=proc\_name |
| handle\_gui\_msgs  (מופעל כ thread ) | gui\_queue – הודעות מהמשתמש דרך הגרפיקה | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  בונה הודעה ע"פ פרוטוקול  ושולחת את ההודעה לשרת |
| create\_socket\_upload\_file | Params - רשימה של משתנים שהפעולה משתמשת בהם | יוצר עצם של clientcomm  client\_upload\_comm  עם הפורט  קורא לפעולה Get\_part\_of\_file עם -1 והשם של הקובץ  קורא לפעולה clientprotocol.Upload\_file  וקורא לפעולה client\_upload\_comm  .send |
| p2p\_download | Params - רשימה של משתנים שהפעולה משתמשת בהם | שומר את הקובץ torrent  יוצר רשימה באורך מספר החלקים עם value -0  יוצר queue של הודעות  קורא לפעולה שמתחילה את הגרפיקה של התקדמות של הקבלת הקובץ  קורא כthread לפעולה get\_part\_from\_queue  ויוצר לכל אחד מהip clientcomm  נקראת הפעולה find\_first\_  ושולח בקשת חלק מכל אחד מהip |
| Build\_com מופעל כ process | Main\_queue – התור שמחובר לmain ומקבל מספר של חלק קובץ לבקש  File\_queue – התור שמחובר את הmain והתקשורת  Ip - הip של השרת  File\_name – שם הקובץ  Len\_of\_part – אורך הקובץ | הפעולה יוצרת קשר p2p עם שרת שיש אצל לקוח שיש לו את הקובץ מוציאה מהתור מספר של חלק מהקובץ ומבקש אותו מהשרת  נסגר ב-1 |
| \_get\_parts\_from\_queue | Comms – מילון שמחזיר רשימה של כל הcomms הפתוחים ששולחים חלקי קובץ  - List\_of\_hashרשימה של הhash של כל החלקים  Full\_data\_hash – הhash הכללי של הקובץ  File\_queue – התור שמקבל את חלקי הקובץ מהלקוחות  File\_name – שם הקובץ  List\_of\_pieces – רשימה שמחזיקה סטטוס על כל חלק קובץ  Len\_of\_part – גודל חלק בקובץ | קורא לפעולה \_Build\_file  כthread מקבל את חלקי הקובץ מהתור, מוודא שהחלק הגיע במלואו משנה את הסטטוס של החלק, מוצא את החלק הבא לבקש בעזרת הפעולה \_Find\_first  במידה והקובץ הגיע כמו שצריך מוסיף לתור את הקובץ בסיום פעולה סוגר את תצוגת ההתקדמות ואת כל התקשורת p2p ואת הפעולה build\_file |
| Build\_file | Build\_queue – התור שאליו מגיע חלקי הקובץ  Path – הדרך לקובץ  Len\_of\_part – גודל החלק | פותח את הקובץ ומחכה שיגיע חלק, מכניס את החלק לתוך הקובץ, כאשר מגיע -1 סוגר את הקובץ |
| \_find\_first | List\_of\_parts- רשימה של סטטוסים של כל חלק  Cant\_recv – מספר חלק שהלקוח הספציפי לא מצליח לקבל  -Number\_foundחלק הקובץ שנשמר | מחזיר את המיקום של החלק הבא למצוא   1. במידה וכל החלקים נמצאו 2. במידה והלקוח לא מצליח להביא את החלק האחרון שנותר |
| Create\_list\_of\_files | Params – רשימת משתנים שהפעולה משתמשת | שולח לgui שיעדכן את רשימת הקבצים |
| Get\_response\_server | Params – רשימת משתנים שהפעולה משתמשת | הפעולה מקבלת את תשובת השרת לאחר העלאת קובץ  במידה והקובץ לא קיים הפעולה שומרת את הקובץ בתיקיית הניטור  אחרת הפעולה מודיע למשתמש בעזרת פעולה בגרפיקה |

| **mainClient** | | |
| --- | --- | --- |
| מחלקה שמאגדת את פעולות למימוש הלוגיקה בשרת הראשי | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| nitur | עצם של הניטור | |
| Nitur\_queue | טור הניטור | |
| Nitur\_comm | עצם תקשורת הניטור | |
| nitur\_comm\_queue | התור שמקבל את ההודעות התקשורת לגבי הניטור | |
| P2p\_server | עצם של השרת הודעות הp2p | |
| P2p\_server\_queue | התור של ההודעות שמתקבלות בp2p | |
| nitur\_commands | מילון שמאגד את הפעולות הניטור ע"פ פעולה - opcode | |
| P2p\_server\_commands | מילון שמאגד את הפעולות שרת הp2p ע"פ פעולה - opcode | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| main |  | יוצר את התורים של הניטור והP2P  יוצר כclientcomm  את עצם תקשורת הניטור  ויוצר עצם monitoring  ויוצר כservercomm את שרת ההודעות הp2p  ומפעיל את הפעולות  Handle\_p2p\_msgs  Handle\_nitur\_comm\_msgs  Handle\_nitur\_msgs |
| Handle\_p2p\_msgs  כthread) | - P2p\_upload\_queue התור שמקבל את ההודעות מלקוחות שמורידים את הקובץ | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  ומפעיל את פעולה send\_part\_of\_file  אם הפרוטוקול מתאים |
| Send\_part\_of\_file | Params – רשימת משתנים שהפעולה משתמשת | בונה הודעה ע"פ פרוטוקול ושולח לו את המידע |
| Handle\_nitur\_comm\_msgs  כthread) | nitur\_comm\_queue – התור שמקבל את ההודעות הניטור מהשרת | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  ומפעיל את הפעולה delete\_file  אם הפרוטוקול מתאים |
| Handle\_nitur\_msgs  (כthread) | Nitur  Nitur\_queue | מוציאה הודעות מהתור מפרקת ע"פ פרוטוקול  מפעילה פעולה רלוונטית  nitur\_commands["opcode"]=proc\_name |
| Delete\_file | - File\_nameשם הקובץ | קורא לפעולה clientfile.delete |
| Added\_file | - File\_nameשם הקובץ | בונה הודעה ע"פ פרוטוקול  שולח את העדכון בעזרת הnitur\_comm.send |
| Changed\_file | - File\_nameשם הקובץ | בונה הודעה ע"פ פרוטוקול  שולח את העדכון בעזרת הnitur\_comm.send |
| Removed\_file | - File\_nameשם הקובץ | בונה הודעה ע"פ פרוטוקול  שולח את העדכון בעזרת הnitur\_comm.send |

| **monitoring** | | |
| --- | --- | --- |
| מחלקה ליצירת עצם שמנטר את התיקייה | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| path | הדרך לתיקייה | |
| msgs\_queue | תור שאליו נכנסות ההודעות | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| \_\_\_init\_\_\_ | Path - הדרך לתיקייה  msgs\_queue - תור שאליו נכנסות ההודעות | יוצר את התיקייה אם לא קיימת ומשנה לה את הסטטוס לhidden  מפעילה thread שפועל על  הפעולה \_monitoring\_folder  שולח את כל הקבצים שקיימים  ומפעיל את הפעולה first change |
| \_monitoring\_folder |  | מנטר את התיקייה ומעלה את השינויים לתור ע"פ פרוטוקול |
| First\_change |  | עושה את השינוי הראשון ומאפס את התיקייה |

| **Clientcomm** | | |
| --- | --- | --- |
| מחלקה שנוצר ממנה מופע על כל סוקט שמתחבר כלקוח | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| Server\_ip | הכתובת של השרת | |
| port | הפורט של הסוקט­ | |
| Message\_Queue | תור שאליו נכנסות ההודעות | |
| Client\_socket | הסוקט שמתחבר לשרת, נוצר בפעולה init | |
| Crypt\_object | אוביקט שמצפין ומפענח | |
| Zfill\_number | המספר בתים של האורך של ההודעה | |
| timer | הזמן לחכות לקבלת קובץ הזמן ההתחלתי הוא 1000, משתנה רק במידה וזה חיבור של p2p | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | Server\_ip – הip של השרת  Port – הפורט של הסוקט  messageQueue - התור שאליו נכנסים ההודעות  zfill\_number – מספר הבתים של האורך של ההודעה | ונקראת הפעולה Recv\_messages\_ כthread |
| recv\_messages\_ |  | נוצר סוקט עם הIp והפורט  מתחבר לשרת  נקראת הפעולה xchange\_key\_ כל הודעה שנכנסת מפוענחת נכנסת לqueue |
| Recv\_file\_ | Params – רשימת משתנים שהפעולה משתמשת | מקבלת את הקובץ בלולאה ע"פ האורך הנדרש  הכנסה לתור המתאים את שם הקובץ, IP, bytearray |
| xchange\_key\_ |  | מחליפה מפתחות, יוצרת עצם הצפנה crypto\_object |
| Send | Message | שולח את ההודעה מוצפנת עם האורך שלה לא מוצפן |
| Send\_file | Data , header | מחכה עד שנוצר עצם של הצפנה ושולח את ההודעה מוצפנת ואת האורך |
| close\_socket |  | סוגר את הסוקט |

### מודולים משותפים

| **Servercomm** | | |
| --- | --- | --- |
| מחלקה שמנהל את התקשורת של השרת, בנוסף נמצאת בכל לקוח כל לאפשר ללקוחות אחרים להתחבר, שרת מרובה לקוחות. | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| Port | הפורט של הסוקט | |
| Message\_Queue | תור שאליו נכנסות ההודעות | |
| Server\_socket | הסוקט של השרת, נוצר בפעולה recv\_messages | |
| Open\_clients | Open\_clients[socket] = [ip,cryptObject] | |
| Zfill\_number | המספר בתים של האורך של ההודעה | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | Port – הפורט של הסוקט  Message\_Queue - התור שאליו נכנסים ההודעות  zfill\_number – מספר הבתים של האורך של ההודעה | נוצר המילון  ונקראת הפעולה Recv\_messages\_ כthread |
| Recv\_messages\_ |  | נוצר סוקט עם הIp והפורט,  כל הודעה שנכנסת מפוענחת נכנסת לqueue עם הip  כאשר לקוח חדש נכנס נקראת הפעולה xchange\_key\_  כthread |
| Xchange\_key\_ | New\_client – הסוקט  Addr - הip | מחליפה מפתחות, יוצרת עצם הצפנה ומעדכנת את המילון |
| find\_socket\_by\_ip\_ | ip | Socket |
| recv\_file\_ | file\_name  client  file\_length | מקבלת את הקובץ בלולאה ע"פ האורך הנדרש  הכנסה לתור המתאים את שם הקובץ, IP, bytearray |
| Send | Message – ההודעה  - ipלמי לשלוח | שולח את ההודעה לאחר הצפנה עם האורך שלה לא מוצפן לסוקט שהip שלו |
| Send\_file | data – המידע של הקובץ  Header – המידע על הקובץ  Ip – הip של הלקוח | שולח ללקוח את הקובץ עם המידע עליו מוצפן ועם האורך |
| Send\_all | message | שולח את ההודעה מוצפנת לכל הלקוחות המחוברים |
| Close\_socket |  | סוגר את הסוקט |

| **Encryption\_decryption** | | |
| --- | --- | --- |
| מחלקה שמייצרת את עצם ההצפנה עם הפעולות | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| key | מפתח ההצפנה | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | key | יוצרת עצם להצפנה סימטרית עם המפתח שהתקבל |
| Encrypt | message | ההודעה מוצפנת ע"י המפתח הסימטרי |
| decrypt | encrypt\_message | Message - ההודעה מפוענחת |
| pad\_ | message | ההודעה לאחר הוספת מידע |
| unpad\_ | Message | ההודעה לאחר הורדת מידע |
| Get\_dif\_num |  | **פעולה חיצונית** – מחזירה 2 מספרים ע"פ דיפי הלמן |
| Set\_key | a,b | **פעולה חיצונית** – מייצרת עצם הצפנה עם המפתח |
| hash | message | **פעולה חיצונית** - ההודעה לאחר גיבוב |

| **settingCli** | |
| --- | --- |
| מחלקה ההגדרות של הפרויקט – משתני הלקוח | |
| **משתני המחלקה** | |
| שם משתנה | תפקיד |
| Time\_to\_connect\_to\_p2p | הזמן להתחבר לp2p |
| Server\_ip | הip של השרת |
| General\_port | הפורט של התקשורת הכללית עם השרת |
| Nitur\_port | הפורט של התקשורת של הניטור עם השרת |
| P2p\_port | הפורט של תקשורת הp2p |
| Bytes\_per\_second | כמות הבתים בשנייה שלקוח מקבל מלקוח אחר |
| Path\_to\_save\_files | המיקום של איפה לשמור קבצים שמשתמש הוריד |
| Nitur\_folder | מיקום של תיקיית הניטור |
| Images\_path | המיקום של תמונות הגרפיקה |

| **settingSer** | |
| --- | --- |
| מחלקה ההגדרות של הפרויקט – משתני השרת | |
| **משתני המחלקה** | |
| שם משתנה | תפקיד |
| General port | הפורט של תקשורת הכללית |
| Nitur\_port | הפורט של תקשורת הניטור |
| Path\_of\_torrent\_folder | הדרך לקבצי הטורנט |
| Len\_of\_part\_of\_file | אורך של חלק מקובץ |

| **graphics** | | |
| --- | --- | --- |
| מחלקת הגרפיקה – ממשק המשתמש | | |
| **משתני המחלקה** | | |
| שם משתנה | תפקיד | |
| File\_list | רשימת הקבצים שמשתמש יכול להוריד עם מידע על כל אחד מהם | |
| total\_parts | כמות החלקים הכוללת של הקובץ שיורד | |
| total\_parts\_download | כמות החלקים שירדו כרגע לקובץ | |
| name\_of\_file | שם הקובץ | |
| **פעולות במחלקה** | | |
| שם פעולה | טענת כניסה | טענת יציאה |
| \_\_init\_\_ | title – הכותרת queue – התור ללוגיקה logo\_path – הדרך לתמונות | יוצר את המסך הראשי ואת כל הכפתורים ומאפס את כל המשתנים |
| After\_upload |  | מראה הודעה הצלחה שהעלאה לשרת עבדה |
| Show\_info\_dialog | Event - | מציג את המידע על הפרויקט |
| File\_exists\_error |  | מציג הודעה שגיאה שהקובץ שהעלאת קיים במערכת |
| On\_upload | Event - | לאחר שנלחץ כפתור העלאה נפתח מסך הקבצים (file explorer) |
| On\_download | Event - | במידה למשתמש יש את הקובץ נשאלת השאלה האם לעביר לו את הקובץ לתיקיית ההורדות אחרת שולח הודעת ללוגיקה ומבטל את כפתור ההורדה. |
| Open\_progress | Total\_part – כמות החלקים  Name\_of\_file – השם של הקובץ | פותח את חלון ההתקדמות של הורדת הקובץ |
| Close\_progress |  | סוגר את חלון ההתקדמות של הורדת הקובץ במידה והגיעו כל החלקים יוצאת הודעת הצלחה שהקובץ ירד אחרת יורדת הודעה שהקובץ לא ירד  מתאפס כל המשתנים |
| Show\_download\_progress |  | מוסיף 1 לסכום החלקים שירדו, בודק האם נלחץ כפתור הביטול במידה וכן מודיע ללוגיקה |
| Update\_file\_list | - New\_file\_listרשימת הקבצים החדשה | משנה את רשימה הקבצים ומציג אותה למשתמש |
| resetdownload |  | מאפס את המשתנים |

1. תיאור סביבת הפיתוח

* שפת התכנות שנבחרה לכתיבת הפרויקט – python 3.7
* עבודה מול בסיס הנתונים – שפת השאילתות SQL
* פירוט כלי הפיתוח והבדיקות הנדרשים לפיתוח
  + PyCharm בשביל כתיבת הקודDebug
  + DB Browser – בדיקת בסיס הנתונים
  + Wireshark - בדיקה מה עובר ברשת ווידוי קבלת הקבצים וההודעות
  + procmon - בדיקת התוכניות הפתוחות

### תיאור האלגוריתמים המרכזיים בפרויקט:

סוגיה 1 – תשתית התקשורת מול השרת בפרויקט

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | כל התקשורת (בקשות, רשימות קבצים, קבצים וכו')עוברת דרך השרת – סוקט יחיד | טיפול בסוקט יחיד – פשוט | יוצר תהליך סדרתי שמעכב את כל התהליכים  נדרש מיון של המידע |
| 2 | הפרדה:  סוקט ראשי  סוקט קבצים | הפרדה בין הקבצים ושאר המידע  פחות איטי | טיפול יותר מורכב  בסוקט הראשי עדיין נדרש מיון של המידע שעובר |
| 3 | הפרדה:  סוקט ראשי  סוקט קבצים  סוקט ניטור | הפרדה בין הקבצים, לבקשות שונות ולמידע על רשימת הקבצים במערכת  אין עיכובים בין סוגי מידע שונים | יותר סוקטים – יותר עבודה  עדיין מכיוון שקיים רק סוקט יחיד להעלאת קבצים – בצד שרת יהיה עיכוב |
| 4 | הפרדה:  סוקט ראשי  סוקט קבצים – לכל לקוח עבור העלאת קבצים  סוקט ניטור | הפרדה בין הקבצים, לבקשות שונות ולמידע על רשימת הקבצים במערכת  אין עיכובים בין סוגי מידע שונים  אין תלות בין לקוחות שמעלים קבצים לשרת | יותר סוקטים – יותר עבודה |

בפרויקט שלי בחרתי בחלופה מספר 4 – בפרויקט שלי נושא המהירות הוא חשוב (תשתית להורדה מהירה של קבצים) ולכן זוהי חלופה נבחרת, ובכתיבה נכונה ברבה סוקטים זו לא בעיה (כפי שניתן לראות בשרטוט הכולל בסעיף 3)

סוגיה 2 – דרך העברת הקבצים

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | התקשורת שדרכה עוברים הקבצים היא השרת, לקוח שרוצה להוריד קובץ מבקש מהשרת את הקובץ והשרת שולח אחרי שביקש מהלקוח האחר | השרת מהווה סמכות ומבטיח שהקובץ המבוקש אכן יגיע ממי שנדרש | איטי ומעמיס על השרת ויותר מזה לא יאפשר הורדת קבצים מהירה  השרת יהווה צוואר בקבוק |
| 2 | בעזרת P2P בין לקוחות לאחר קבלת קובץ הtorrent מהשרת | מהירות ועבודה יעילה מוריד הרבה עומס מהשרת | נדרשת דרך לאמת את הקובץ |

בפרויקט שלי בחרתי בחלופה 2- בפרויקט שלי חושב המהירות והעבודה היעילה והורדת הצוואר בקבוק מהשרת, בנוסף לזה יש לי דרך לאמת את הקובץ בעזרת השימוש בhash

סוגיה 3 – ניהול רשימת הקבצים במערכת (מערכת ניטור)

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | השרת מבקש כל זמן מסוים את הרשימה מהלקוחות | מאפשר שליחה של כל השינויים שהתרחשו בפרק זמן מסוים במקום שליחה של כל אחד בנפרד | קבלת העדכון ברגע בקשת השרת ולא ברגע השינוי |
| 2 | הלקוחות מנטרים את תיקיית הקבצים ושולחים כל שינוי לשרת | שם את עבודת העדכון על הלקוחות ומוריד לחץ מהשרת  קבלת הודעה על העדכון באותו רגע של העדכון | מוסיף עבודה לכל לקוח (יצירת הניטור) |

בפרויקט שלי בחרתי בחלופה 2 – בפרויקט שלי חשוב המהירות של עדכון הלקוחות לגבי רשימת הקבצים האפשריים להורדה, חלופה 2 נותנת את הדרישה

סוגיה 4 – מימוש ה Hash בפרויקט

| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | חישוב ובדיקת hash כולל לקובץ | חוסך בביצוע hash | במידה והקובץ אינו תקין הלקוח יגלה על כך רק בסיום החיבור ולכן הפתרון יכול להיות מאוד לא יעיל מהיר |
| 2 | חישוב ובדיקה hash לכל חלק בקובץ | חוסך בביצוע hash כולל, חוסך חיבור של כל הקובץ במידה וחלק אחד לא תואם | חישוב hash של חלק קטן יכול לצאת שווה לחלק אחר, לכן יכול להיות הצגה של חלק קובץ תקין כלא תקין וההפך, מה שיפגע בכל הקובץ |
| 3 | חישוב ובדיקת hash כולל וגם חישוב ובדיקת hash לכל חלק | חוסך חיבור של כל הקובץ במידה וחלק אחד לא תואם, מאפשר בדיקה נוספת ווידוי נוסף בסיום חיבור הקובץ | דורש הרבה חישוב ובדיקות hash |

בפרויקט שלי בחרתי בחלופה 3 – בפרויקט שלי יש דגש על ווידוי הגעה של הקובץ הנכון ללקוח בנוסף לזה חשוב לי לייעל את הלקוח ולחסוך ממנו עבודה מיותרת, חלופה 3 נותנת את הפתרון המתאים ביותר.

סוגיה 5 – סוג ההצפנה בפרויקט

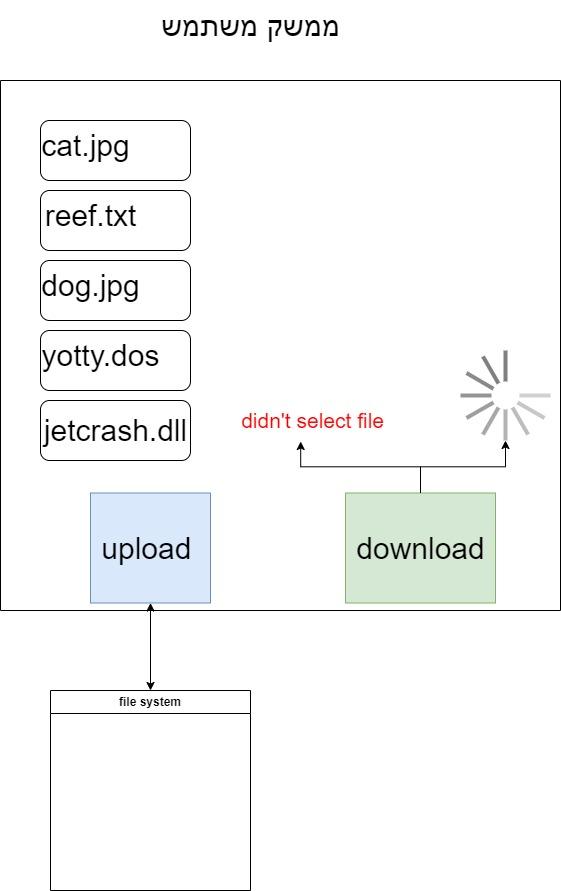
| חלופה | תיאור | יתרונות | חסרונות |
| --- | --- | --- | --- |
| 1 | שימוש בהצפנה סימטרית | נותן מהירות בהצפנה והפענוח | המפתח עובר ברשת |
| 2 | שימוש בהצפנה א-סימטרית | המפתח לא עובר ברשת | איטי יותר |
| 3 | שילוב של ההצפנות שימוש בדיפי הלמן לקביעת המפתח מבלי להעביר את המפתח עצמו ברשת, הצפנה סימטרית עם המפתח שנקבע | המפתח לא עובר ברשת, נותן מהירות | דורש ביצוע של 2 חלקים |

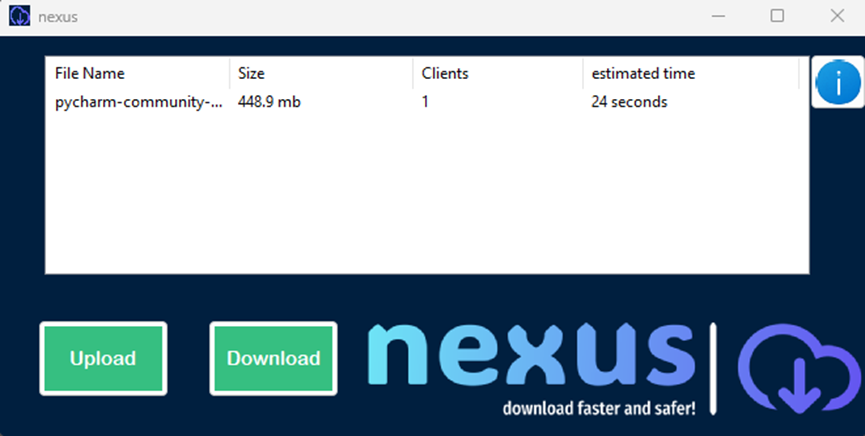
בפרויקט שלי בחרתי בחלופה 3 – בפרויקט שלי יש גם דגש על ביטחון המידע וגם על מהירות, בעזרת חלופה 3 אני גם לא צריך להעביר את המפתח ברשת מה שמשפר את ביטחון המידע וגם אני משתמש בהצפנה סימטרית כדי לאפשר מהירות.

### תיאור מסכי הפרויקט:

* + מה מטרת המסך – להציג למשתמש את הקבצים שהוא יכול להוריד ולאפשר לו לעלות קבצים למערכת

**סקיצה**





כאשר נלחץ כפתור העלאה נפתחת מערכת הקבצים והמשתמש מתבקש לבחור קובץ

כאשר נלחץ כפתור ההורדה:

* במידה ולא נבחר קובץ נשלחת הודעת שגיאה
* במידה וכן נבחר קובץ נפתחת הצגה של טעינה של הורדת הקובץ

### תיאור פרוטוקול התקשורת

**פרוטוקול סוקט תקשורת רגילה**

אורך הודעה (מוצפנה) – 4 בתים

הודעה מוצפנת

| הודעה | מ- | אל | opcode | שדות | דוגמא |
| --- | --- | --- | --- | --- | --- |
| העברת מפתחות(דיפי הלמן) | לקוח | שרת | 00 | A-מספר | 0012341321 |
| העברת מפתחות (דיפי הלמן) | שרת | לקוח | 00 | B-מספר | 0012367521 |
| בקשת הקובץ torrent | לקוח | שרת | 01 | שם קובץ | 01cat.jpg |
| תשובת בקשת קובץ torrent | שרת | לקוח | 01 | 0 – שגוי  או  1  ואת הקובץ  קובץ ה torrent | 010  או  011b'0x1a4b 1cd5 ff23' |
| בקשת העלאה | לקוח | שרת | 02 | שם קובץ | 02Cat.jpg |
| תשובה לבקשה העלאה | שרת | לקוח | 02 | 0 – קיים  או  1 - שם הקובץ פורט 4 ספרות | 020  0211500Cat.jpg |
| עדכון רשימת הקבצים | שרת | לקוח | 03 | שמות הקבצים | 03Cat.jpg$%$dog.jpg |

**פרוטוקול סוקט ניטור:**

אורך הודעה(מוצפנת) 2 בתים

הודעה מוצפנת

| הודעה | מ- | אל | opcode | שדות | דוגמא |
| --- | --- | --- | --- | --- | --- |
| נוסף קובץ | לקוח | שרת | 01 | שם קובץ | 01cat.jpg |
| נמחק קובץ | לקוח | שרת | 02 | שם הקובץ | 02cat.jpg |
| השתנה קובץ | לקוח | שרת | 03 | שם הקובץ | 03cat.jpg |
| הוראת מחיקת קובץ | שרת | לקוח | 03 | שם קובץ | 03Cat.jpg |
| העברת מפתחות(דיפי הלמן) | לקוח | שרת | 00 | A-מספר | 0012341321 |
| העברת מפתחות (דיפי הלמן) | שרת | לקוח | 00 | B-מספר | 0012367521 |

**פרוטוקול סוקט העלאה:**

אורך הודעה(מוצפנת) 8 בתים

הודעה מוצפנת

| הודעה | מ- | אל | opcode | שדות | דוגמא |
| --- | --- | --- | --- | --- | --- |
| העלאת קובץ | לקוח | שרת | 01 | אורך הheader  אורך הקובץ  opcode  שם קובץ  הקובץ | 00233425$%$01$%$ cat.jpg $%$b'0xaf43' |
| תשובה לשליחה | שרת | לקוח | 01 | 1 – עבר | 011 |

**פרוטוקול סוקט לקוח הורדה:**

אורך הודעה(מוצפנת) 4 בתים

הודעה מוצפנת

| הודעה | מ- | אל | opcode | שדות | דוגמא |
| --- | --- | --- | --- | --- | --- |
| העברת מפתחות  (דיפי הלמן) | לקוח | לקוח | 00 | B-מספר | 0012367521 |
| בקשת חלק קובץ | לקוח | לקוח | 01 | מספר חלק - 3 ספרות  שם קובץ | 01007cat.jpg |

**פרוטוקול סוקט שרת הורדה:**

אורך הודעה(מוצפנת) 4 בתים

הודעה מוצפנת

| הודעה | מ- | אל | opcode | שדות | דוגמא |
| --- | --- | --- | --- | --- | --- |
| העברת מפתחות  (דיפי הלמן) | לקוח | לקוח | 00 | A-מספר | 0012341321 |
| שליחת חלק קובץ | לקוח | לקוח | 10 | אורך הheader  opcode  שם הקובץ  אורך הdata  מספר החלק – 3 ספרות  החלק עצמו | 2201cat.jpg$%$datalen$%$007B'0xa0f2 55fa' |

### תיאור מבני הנתונים

**שרת**

בסיס נתונים – nexusDB

טבלה torrentTBL - מכילה שמות של קבצים הtorrent הקיימים (שם הקובץ הוא המפתח – לא יתכנו שני קבצים עם אותו השם)

| Names |
| --- |
| STRING עד 30 |
| Cat.jpg.torrent |
| Clients.py.torrent |
| Imri.txt.torrent |

תיקיית קבצים – Torrent files

מכילה את קבצי הtorrent

רשימת לקוחות מחוברים-open\_sockets\_clients

שומר socket – ip, crypt\_object לכל לקוח שמחובר למערכת

רשימת קבצים – open\_file\_names

שומר את שמות הקבצים השמורים אצל הלקוחות המחוברים כרגע למערכת, מתעדכן כל רגע נתון

**מבנה קובץ Torrent - Json**

| **דוגמא** | **value** | **Key** | **הסבר** |
| --- | --- | --- | --- |
| Cat.jpg | String | File name | השם של הקובץ |
| ["192.168.4.97","192.168.4.96"] | List of strings | Open IP's | כל הip של הלקוחות שיש להם את הקובץ |
| 25600000 | Int | Length of file (bytes) | האורך של הקובץ בבתים |
| 1000 | Int | Number of pieces | מספר החלקים שיש לקובץ |
| ["0xa878b765a","0x1626ab71ab7"] | String | Hash of each piece in order | הhash של כל לקוח |
| 0x123b1a1b3c131"" | String | Full hash | הhash של כל הקובץ |

**לקוח**

תיקייה קבצים מנוטרת – Nexus(hidden)

מכילה קבצים שהמשתמש מאפשר לשתף עם משתמשים אחרים

רשימת קבצים – open\_files\_name

שומר את שמות הקבצים השמורים אצל הלקוחות המחוברים כרגע למערכת, מתעדכן כל רגע נתון

### סקירת חולשות והאיומים

|  |  |  |
| --- | --- | --- |
| האיום | תיאור | פתרון |
| DOS | לקוח יחיד מתחבר לשרת ומתנתק מספר פעמיים גדול מספיק בשביל לתקוע את השרת | השרת אינו יאפשר התחברות מאותו הip מספר בלתי מוגבל של פעמיים |
| DDOS | לקוחות רבים מתחברים לשרת ומתנתקים | הפרויקט אינו מציג פתרון |
| SQL injection | כתיבת קלט שהוא שורות קוד לתוך הבסיס נתונים | כתיבת שאילתה נכונה, שלא מאפשרת הכנסת קוד זדוני |
| Man in the middle | אדם עם כוונות רעות מאזין להודעות העוברות ומתחיל להתחזות לאחד מהגורמים המתקשרים | כל התקשורת מוצפנת, גם אם האדם מאזין הוא אינו יכול להבין מה משמעות ההודעות, שימוש בפרוטוקול tcp בשכבת התעבורה עם לחיצת יד משולשת |
| קובץ לא תקין | שליחה של לקוח ללקוח אחר קובץ שאינו תקין | השוואת הhash של חלק הקובץ ושל הקובץ הכולל עם הhash המאומת על ידי השרת |
| וירוס | העלאת קובץ שמכיל וירוס | הפרויקט אינו בודק את תוכן הקובץ |

## פרק ה' - nexus – הקוד

### קטעי קוד מעניינים

**סוגיה 1** – תשתית התקשורת מול השרת בפרויקט

*# the general comm*general\_comm = ServerComm.ServerComm(settingSer.GENERAL\_PORT, general\_queue, 6)

*# the nitur comm*nitur\_comm = ServerComm.ServerComm(settingSer.NITUR\_PORT, nitur\_queue, 2)

upload\_comm = ServerComm.ServerComm(port, upload\_queue, 8)

**סוגיה 2** – דרך העברת הקבצים

יצירת התקשורת p2p

comm = ClientComm.Clientcomm(ip, file\_queue, settingCli.P2P\_PORT, 4)

**סוגיה 3** – ניהול רשימת הקבצים במערכת (מערכת ניטור)

התיקייה מנוטרת ומודיעה על כל שינוי לשרת

**def** \_monitoring\_folder(self):  
 *'''  
 monitor a directory and record the changes in a file and on the screen* **:return***:  
 '''  
 # set up the notification handle* change\_handle = win32file.FindFirstChangeNotification(  
 self.path\_to\_monitor,  
 0,  
 win32con.FILE\_NOTIFY\_CHANGE\_FILE\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_DIR\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_SIZE  
  
 )  
 **try**:  
 **while True**:  
 result = win32event.WaitForSingleObject(change\_handle, win32event.INFINITE)  
 *# check if there are any changes* **if** result == win32event.WAIT\_OBJECT\_0:  
 **for** action, filename **in** win32file.ReadDirectoryChangesW(change\_handle, 1024, **True**,  
 win32con.FILE\_NOTIFY\_CHANGE\_FILE\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_DIR\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_LAST\_WRITE):  
 *# add file* **if** action == 1:  
 self.list\_of\_new\_files.append(filename)  
 *# delete file* **elif** action == 2:  
 self.msgs\_queue.put((**"02"**, filename))  
 *# change file* **elif** action == 3:  
 **if** filename **in** self.list\_of\_new\_files:  
 self.list\_of\_new\_files.remove(filename)  
 self.msgs\_queue.put((**"01"**, filename))  
 **else**:  
 self.msgs\_queue.put((**"03"**, filename))  
 *# change file* **elif** action == 4:  
 self.msgs\_queue.put((**"03"**, filename))  
 *# change name file* **elif** action == 5:  
 self.msgs\_queue.put((**"05"**, filename))  
 **except** Exception **as** e:  
 win32file.FindCloseChangeNotification(change\_handle)

**סוגיה 4** – מימוש ה Hash בפרויקט

עבור כל קובץ torrent מחושב hash כללי וגם לכל חלק

torrent\_file = self.\_build\_torrent\_file\  
 (data, filename, full\_hash, list\_of\_hash, settingSer.len\_of\_part\_of\_file)

בדיקת hash של חלק

hash\_part = Encryption\_Decryption.AES\_encryption.hash(data)  
*# check hash  
# bad hash***if** list\_of\_hash[number\_of\_part] != str(hash\_part):

בדיקת hash כולל

full\_hash = Encryption\_Decryption.AES\_encryption.hash(data)  
*# check the full hash***if** full\_data\_hash == str(full\_hash):

**סוגיה 5** – סוג ההצפנה בפרויקט

מגיע למפתח בעזרת דיפי הלמן

**def** get\_dif\_Num():  
 *"""* **:return***: 2 numbers in dif format  
 """* a = random.randint(1, p)  
 **return** a, str((g\*\*a) % p)

משתמש בהצפנה סימטרית AES

**class** AES\_encryption(object):  
 **def** \_\_init\_\_(self, key):  
 *"""* **:param** *key: the key  
 """* self.bs = AES.block\_size  
 self.key = hashlib.sha256(str(key).encode()).digest()

## פרק ו' – nexus – בדיקות ('קופסא לבנה')

### פירוט הבדיקות:

| **מספר** | **שם הבדיקה** | **מה אמורה לבדוק** | **איך מתכננים לבדוק** | **תאריך** | **מה בוצע בפועל** |
| --- | --- | --- | --- | --- | --- |
| 1 | הורדת קובץ ממחשב אחד | שמחשב מצליח להוריד קובץ | לחבר 2 לקוחות שלאחד מהם אין קובץ מסוים לאחד מהם יש, ולנסות להוריד את הקובץ למחשב שאין לו את הקובץ בסיום ההורדה לפתוח את הקובץ ולבדוק שהוא זהה למקור. | 4.4.2024 | הורדת קובץ קטן – הקובץ ירד והופיע בהורדות  הורדת קובץ גדול – הלקוח קרס, הבעיה הייתה כשאר עשיתי recv(1024) לא בהכרח קיבלתי 1024 בתים, טיפלתי בזה בכך שבדקתי כמה בתים קיבלתי |
| 2 | הורדת קובץ מכמה מחשבים | שהמחשב מוריד את הקובץ במהירות גדולה יותר בעזרת כמה מחשבים | לחבר כמה מחשבים שלכולם יש את אותו הקובץ ואז לחבר מחשב נוסף שאין לו את הקובץ ולבקש להוריד את הקובץ  ולמדוד את הזמן שלקח לו להוריד את הקובץ, להשוואת את הזמן לזמן שלוקח ללקוח להוריד את אותו הקובץ בעזרת מחשב אחד | 4.4.2024 | קובץ קטן < 1024kb – יש שינוי קטן מאוד בזמנים יכול להיות קצת יותר או קצת פחות תלוי במחשב ובעומס הרשת  קובץ גדול > 1024kb הקובץ ירד יותר מהר ככל שהיו יותר מחשבים |
| 3 | התרעת על קובץ לא שלם | שהמערכת מתריעה ללקוח שחלק מהקובץ או הקובץ כולו שהוא הוריד ממחשב מסוים לא תואם את הקובץ שהוא היה אמור להוריד | לחבר 2 מחשבים לרשת לבקש להוריד קובץ מסוים, המחשב שמורידים ממנו את הקובץ צריך לערוך את הקובץ שהוא לא יהיה שלם, ולבדוק האם המערכת מצליחה לזהות את הבעיה. | 4.4.2024 | לקוח ניסה להוריד קובץ, תוך כדי ההורדה המשתמש שעל הלקוח עם הקובץ שינה אותו באמצע, המשתמש שניסה להוריד, קיבל הודעת שגיאה אך הלקוח עם הקובץ קרס, לאחר תיקון הלקוח אינו קורס והמערכת מתעדכנת, לא ניתן למחוק קובץ פתוח, יתוקן במידה ויהיה זמן |
| 4 | בדיקת פירוק ואיחוד קובץ | שיש הצלחה לפרק קובץ לחלק אותו לכמה מחשבים, ואז לקבל את החלקים מהמחשבים חזרה ולחבר | לחבר 2מחשבים ולהוריד את קובץ ולבדוק את הזמן שלוקח לו להוריד  לאחר מכן לפתוח מחשב נוסף להוסיף לו את הקובץ, ולמחוק את הקובץ שהורדת בפעם הראשונה אצל המחשב הראשון ואז לבקש שוב להוריד את אותו הקובץ ולבדוק האם לוקח לו פחות זמן העובדה שזמן ההורדה קטן מעידה על זה שהקובץ ירד מכמה מחשבים ושהפירוק והאיחוד עובד | 4.4.2024 | הקובץ מתחבר לאחר הורדה לקובץ שלם |
| 5 | בדיקה של קישור ממשק לקוח למערכת | שאחרי שהלקוח מבקש קובץ מסוים ההודעה עוברת בתקשורת והוא מקבל את הקובץ | לפתוח לקוח ולבדוק שמופיע הממשק משתמש ולבקש ממנו להוריד קובץ ולבדוק האם הקובץ יורד | 4.4.2024 | ממשק המשתמש נפתח ומאפשר למשתמש להוריד קובץ ולהעלות קובץ |
| 6 | עבודה במקביל | שניתן להוריד כמה קבצים במקבלים (כמה משתמשים שונים ) | לפתוח 2 מחשבים ולבדוק כמה זמן לוקח להם להוריד קובץ מסוים, לאחר מכן למחוק את הקובץ ממחשב אחד, ואז לפתוח כמה מחשבים ולבקש להוריד את אותו הקובץ מהפעם הראשונה ובנוסף לזה לבקש קובץ אחר ממחשב אחר ולבדוק שהעבודה במקביל לא משפיעה על הזמן של הורדת הקובץ הראשון | 4.4.2024 | נפתחו כמה מחשבים והקבצים עולים במקביל לשרת בצורה טובה. |
| 7 | הורדה במקבילים | שניתן להוריד את אותו הקובץ במקביל אצל שני לקוחות או יותר | לפתוח כמה מחשבים ולבקש מ2 מהם קובץ שקיים בכמה מחשבים אחרים | 4.4.2024 | הורדה במקביל כשיש 2 לקוחות שיש להם את הקובץ עובד, והזמן קצר יותר מהורדה מלקוח אחד  הורדה במקביל כשיש לקוח אחד שיש לו את הקובץ, הזמן לא מושפע והקובץ יורד טוב |
| 8 | מערכת עדכנית | שלא ניתן להוריד קובץ מהמערכת אם הוא נמחק מהלקוחות שהורידו אותו | לפתוח מחשב ולהוסיף אליו קובץ שקיים רק אצלו ואז לבדוק האם אפשר להוריד את הקובץ לאחר מכן למחוק את הקובץ בכל המחשבים שהקובץ קיים עליהם ולבדוק האם הוא עדיין מופיע במחשב שלישי שלא היה עליו את הקובץ | 4.4.2024 | המשתמש לא רואה קבצים שנמחקו מהלקוחות שהיה להם אותו |

## פרק ז' - nexus – מדריך למשתמש

### מדריך למשתמש (בוחן)

יש לשנות את הsettingCli:

Image\_path – הדרך לקבצים של הגרפיקה

Nitur\_folder – הדרך לתיקייה של הפרויקט, הקבצים של המערכת ישמרו שם, התיקייה נוצרת ע"י המערכת, הדרך צריכה להיות קיימת

Path\_to\_save\_files – המיקום שבו קובץ שירד מהמערכת ישמר

Server\_ip – הip שבו רץ השרת

יש לשנות את הsettingSer:

Path\_of\_torrent\_folder – הדרך לתיקייה שבה נשמרים קבצי הטורנט, הדרך חייבת להיות קיימת, התיקייה נוצרת ע"י המערכת

בצד השרת:

נדרש להריץ את הmainServer , יש לוודא שבסיס הנתונים ריק לפני ההרצה הראשונית

בצד הלקוח:

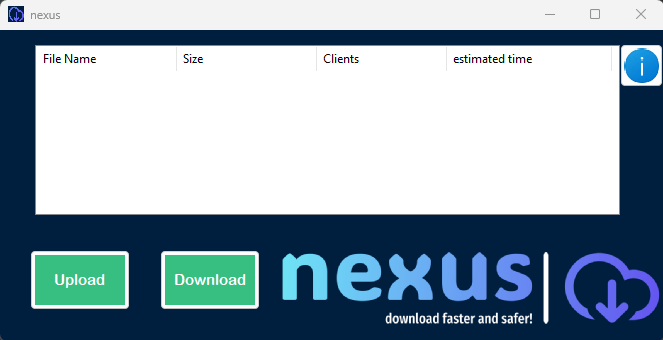
דבר ראשון נדרש להריץ את הmainClient, קוד זה מריץ לקוח שמאפשר להוריד ממנו קבצים.

במידה ונרצה להתחבר כמשתמש נדרש להריץ את הmainClientUser **לאחר** הרצת הmainClient קוד זה יפתח את הגרפיקה ויאפשר למשתמש להוריד ולהעלות קבצים למערכת.

### מדריך למשתמש

יש לוודא שהקובץ mainClient רץ, בנוסף לזה יש להריץ את mainClientUser

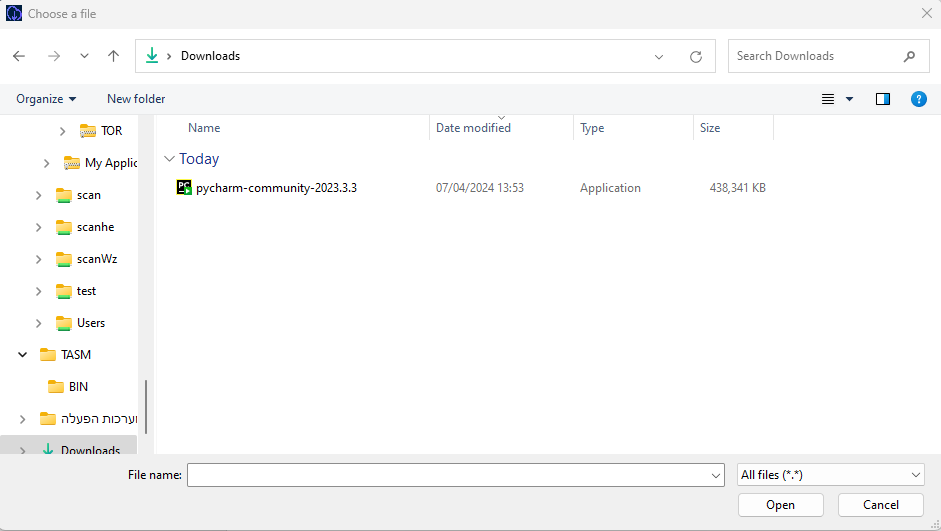
כאשר משתמש פותח את המערכת זה המסך הראשוני שיפתח



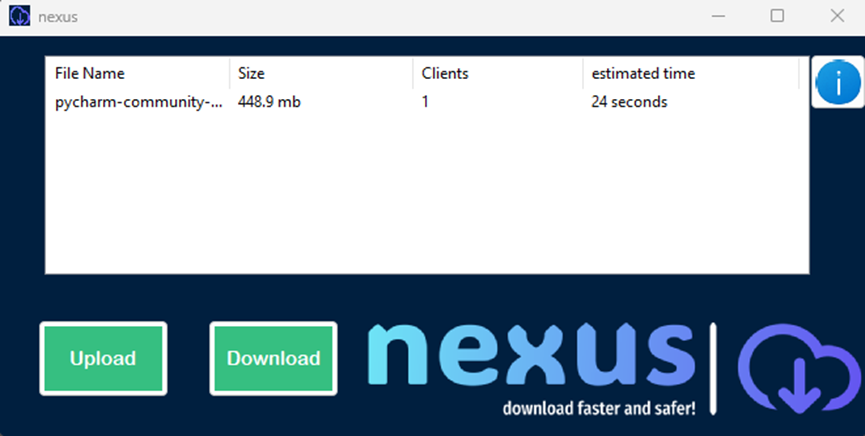
לחיצה על הכפתור הזה יפתח הסבר אודות על המערכת

לחיצה על הכפתור הזה יפתח את חלון הקבצים (file explorer) ויאפשר למשתמש לבחור קובץ ולהעלות אותו למערכת, תוך כדי העלאה הכפתור יהיה אפור.

**חלון הקבצים**



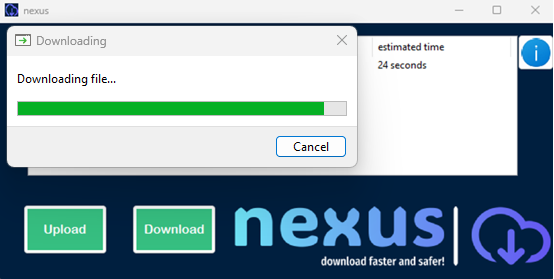
לאחר העלאת הקובץ יוצג במסך של כל אחד מהמשתמשים את שם הקובץ ומידע על עליו כגון לכמה לקוחות יש את הקובץ, גודל הקובץ והזמן ההורדה שלו. (זמן ההורדה מבוסס על הרשת של בית הספר, זמן זה יכול להתקצר ולהתארך תלוי רשת ותלוי מחשב).



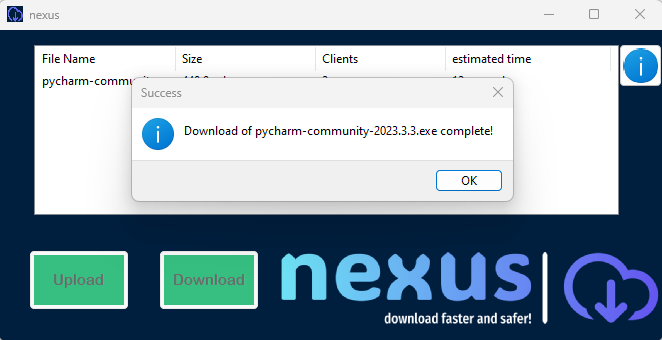
כאשר משתמש מעוניין להוריד קובץ שנמצא במערכת הוא יכול לסמן את הקובץ בעזרת לחיצה על שם הקובץ ולאחר מכן ללחוץ על כפתור ההורדה

כפתור ההורדה

לאחר לחיצה על כפתור ההורדה יפתח סרגל התקדמות שיראה את ההתקדמות תוך כדי הורדת הקובץ.

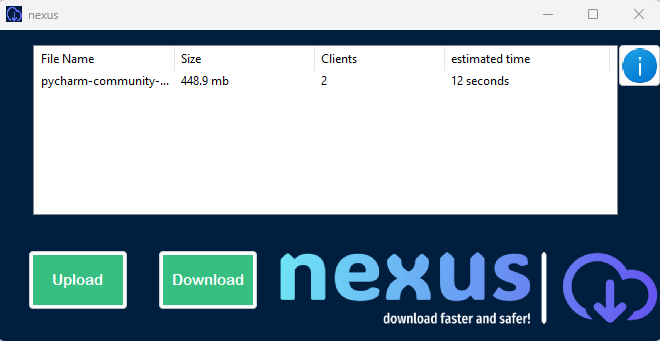


בסיום ההורדה יפתח הודעה של "הקובץ ירד בהצלחה" ויחכה לאישור מהמשתמש



תוך כדי הורדת הקובץ הכפתורים אינם יהיו זמינים והצבע שלהם יהפוך לאפור, בסיום הורדת הקובץ ולאחר לחיצה על אישור, הכפתורים יחזרו לפעול.

" ולאחר כמה שניות המערכת תעדכן את הגרפיקה של כל הלקוחות, הקובץ ישמר במערכת, בנוסף לזה יוצג בתיקיית ההורדות.



המערכת העדכנית לאחר ההורדה, מספר הלקוחות גדל וזמן ההורדה קטן.

## פרק ח' - nexus – רפלקציה

### מבט אישי על העבודה ועל תהליך פיתוחה:

הפרויקט הזה היה חוויה לימודית מרגשת ומאתגרת. שילוב של שעות ארוכות של חיפוש פתרונות ומציאת בעיות מאתגרות ושל שמחה גדולה בעת מציאת הפתרון ופתירת הבעיה.

האתגר הראשון שהיה לי הוא למצוא דרך לממש את הbitTorrent האם ליצור שרת, האם לממש רק את הפרוטוקול של הלקוח הקיים או האם לבנות את שניהם וליצור פרוטוקול חדש משלי. החלטתי לבנות שרת מרכזי ולממש פרוטוקול משלי, התחלתי בחשיבה תאורתי ובכתיבת התיק פרויקט למצוא דרך ממשית לבנות את האלגוריתם של רשת הbitTorrent לאחר מחשבה ארוכה ורבת אתגרים ובעזרת המורה המנחה מרי שעזרה ותמכה לאורך כל הפרויקט ועם קצת התייעצויות עם חברי לכיתה ודיונים ארוכים הגעתי לפתרון שבו הייתי גאה מאוד ומאוד מרוצה מהתוצאה.

אחד האתגרים הממשיים שהיו לי הוא איך לנהל את אופן הבקשות של חלקי הקובץ ומתי להחליט להפסיק לבקש מלקוח מסוים ולהוציא אותו מרשימת הלקוחות שאני מוריד מהם את הקובץ, או איך לדעת איזה חלק קובץ לבקש ממי ומתי. בסופו של דבר לאחר הרבה תיקונים ושיפורים הגעתי לאלגוריתם סופי שממנו אני מאוד מרוצה.

שתי בעיות שהעסיקו אותי לא מעט במהלך הפרויקט היו בעיית המקרי קיצון ובעיה נוספת שנגרמה מזה שכאשר קיבלתי קובץ גדול ברשת וכל פעם קיבלתי 1024 בתים ( (recv(1024)כאשר כתבתי את שורת הקוד הזאת יצאתי מנקודת הנחה שאני בהכרח אקבל 1024 בתים, אך מסקנה זאת הייתה שגויה, דבר זה יצר בעיה שלקחה לי זמן רב למצוא, כאשר מצאתי את הבעיה התיקון היה קצר, במקום להניח שקיבלתי 1024 בתים בדקתי כמה בתים קיבלתי ולפי זה הורדתי מהסכום.

בעיית המקרי קיצון הייתה בעיה לא קלה, שכל פעם שחשבתי עוד קצת על הפרויקט מצאתי בעיית קיצון נוספת. פתירת הבעיות קיצון חייבה אותי לחשוב בצורה יצירתית כדי לפתור וכדי למצוא את אותם הבעיות.

תוך כדי הפרויקט למדתי המון דברים חדשים בין אם זה להבין ולדעת להשתמש בפורמט json להכיר יותר לעומק את ה winapi ללמוד על תיקיות נסתרות ולהבין יותר לעומק את השימושים ומקרי קצה שיש בקבצים מסוימים ואפילו לקרוא על גדלי מידה אופטימליים להעברה ברשת. בנוסף לכל הדברים האלו למדתי גם כמה חשוב לתכנן מראש ולהסתכל קדימה על התוצר הסופי תוך כדי התקדמות איטית ומחושבת שלב לאחר שלב.

לסיום אני רוצה להגיד תודה למרי גבע שתמכה ליוותה ועזרה לי לאורך כל הפרויקט ונתנה לי השראה וכוח להמשיך לעבוד ולהתקדם גם שהיה קשה , ולכל החברים לכיתה שיצרו סביבה לימודית שכיף לעבוד בה ואפשר לנהל בה דיונים ולהתייעץ אחד עם השני על בעיות שעולות.

## פרק ט' - nexus – ביבליוגרפיה

קבצים JSON בפיתון

<https://www.geeksforgeeks.org/read-json-file-using-python/>

מאמר על bit torrent network

<https://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf>

שימוש בwinAPi בשביל ניטור תיקייה

<https://learn.microsoft.com/en-us/windows/win32/api/synchapi/nf-synchapi-waitforsingleobject>

הוספת מידע לאמצע של קובץ

<https://www.geeksforgeeks.org/python-seek-function/>

נספח:

**from** allcode **import** ServerComm  
**from** serverCode **import** ServerFiles  
**from** serverCode **import** DB  
**from** serverCode **import** serverProtocol  
**import** queue  
**import** os  
**from** allcode **import** settingSer  
**import** threading  
**import** multiprocessing  
**from** serverCode.Process\_handle\_file **import** handle\_upload\_file  
  
  
**def** handle\_nitur\_msgs(nitur\_queue):  
 *"""  
 call the func by the opcode* **:param** *nitur\_queue: the nitur queue  
 """* **while True**:  
 ip, message = nitur\_queue.get()  
 opcode, name\_of\_file = serverProtocol.serverProtocol.unpack(message)  
 nitur\_commands[opcode]([name\_of\_file[0], ip])  
  
  
**def** file\_added(params):  
 *"""  
 add the file to the list of file and send to every one  
 :params: list of params  
 """* name\_of\_file, ip = params[0], params[1]  
 torrents\_db = DB.DBClass()  
 have\_torrent = torrents\_db.have\_torrent(name\_of\_file)  
 torrents\_db.closeDb()  
 **if** have\_torrent:  
 *# add ip to torrent file* files\_obj.add\_ip\_to\_torrent(name\_of\_file, ip)  
 *# add to the count of clients of the file* open\_files[name\_of\_file][1] = open\_files[name\_of\_file][1]+1  
 string\_of\_open\_files = serverProtocol.serverProtocol.Create\_string\_of\_list(open\_files)  
 *# send update to every client* general\_comm.sendall(string\_of\_open\_files)  
 **else**:  
 *# send to the client to delete the file from nitur* delete\_file\_msg = serverProtocol.serverProtocol.Delete\_file\_from\_folder(name\_of\_file)  
 nitur\_comm.send(delete\_file\_msg, ip)  
  
  
**def** file\_deleted(params):  
 *"""  
 delete the client ip from the torrent and change the list of files* **:param** *params: list of params  
 """* name\_of\_file, ip = params[0], params[1]  
 torrents\_db = DB.DBClass()  
 have\_torrent = torrents\_db.have\_torrent(name\_of\_file)  
 torrents\_db.closeDb()  
 **if** have\_torrent:  
 *# delete the ip from the torrent* files\_obj.delete\_ip\_from\_torrent(name\_of\_file, ip)  
 **if** name\_of\_file **in** open\_files:  
 *# delete client from the count of open clients* open\_files[name\_of\_file][1] = max(0, open\_files[name\_of\_file][1]-1)  
 string\_of\_open\_files = serverProtocol.serverProtocol.Create\_string\_of\_list(open\_files)  
 *# send every client update* general\_comm.sendall(string\_of\_open\_files)  
  
  
**def** file\_changed(params):  
 *"""  
 delete the client ip from the torrent and change the list of files* **:param** *params: list of params  
 """* name\_of\_file, ip = params[0], params[1]  
 torrents\_db = DB.DBClass()  
 have\_torrent = torrents\_db.have\_torrent(name\_of\_file)  
 torrents\_db.closeDb()  
 **if** have\_torrent:  
 *# delete ip from torrent* files\_obj.delete\_ip\_from\_torrent(name\_of\_file, ip)  
 **if** name\_of\_file **in** open\_files:  
 *# remove the client from the count of clients* open\_files[name\_of\_file][1] = max(0, open\_files[name\_of\_file][1]-1)  
 string\_of\_open\_files = serverProtocol.serverProtocol.Create\_string\_of\_list(open\_files)  
 *# send every client* general\_comm.sendall(string\_of\_open\_files)  
 *# send to the client to delete the file from the nitur file* delete\_file\_msg = serverProtocol.serverProtocol.Delete\_file\_from\_folder(name\_of\_file)  
 nitur\_comm.send(delete\_file\_msg, ip)  
  
  
**def** file\_name\_changed(params):  
 *"""  
 send the client to delete the file from nitur* **:param** *params: list of params  
 """* name\_of\_file, ip = params[0], params[1]  
 delete\_file\_msg = serverProtocol.serverProtocol.Delete\_file\_from\_folder(name\_of\_file)  
 nitur\_comm.send(delete\_file\_msg, ip)  
  
  
**def** handle\_general\_msgs(general\_queue):  
 *"""  
 call the func by the opcode* **:param** *general\_queue: the queue connect to the general comm  
 """* **while True**:  
 ip, message = general\_queue.get()  
 opcode, params = serverProtocol.serverProtocol.unpack(message)  
 params.append(ip)  
 general\_commands[opcode](params)  
  
  
**def** send\_list\_of\_files(params):  
 *"""  
 send the list of files to the client* **:param** *params:list of params* **:return***:  
 """* ip = params[0]  
 message = serverProtocol.serverProtocol.Create\_string\_of\_list(open\_files)  
 general\_comm.send(message, ip)  
  
  
**def** send\_torrent(params):  
 *"""  
 send the torrent file to the client* **:param** *params: list of params  
 """* file\_name = params[0]  
 ip = params[1]  
 *# get the torrent file* torrent\_file = files\_obj.get\_torrent\_file(file\_name)  
 header = serverProtocol.serverProtocol.Response\_for\_torrent\_request(file\_name)  
 general\_comm.send\_file(torrent\_file, header, ip)  
  
  
**def** create\_uplaod\_socket(params):  
 *"""  
 create the upload comm and send to the client the info* **:param** *params: list of params* **:return***:  
 """* path\_of\_file, ip = params[0], params[1]  
 *# file\_name = os.path.basename(path\_of\_file)* torrents\_db = DB.DBClass()  
 torrents\_db.closeDb()  
 upload\_queue = multiprocessing.Queue()  
 list\_queue = multiprocessing.Queue()  
 *# get port from the generator* port = next(unused\_ports)  
 threading.Thread(target=\_after\_upload, args=(list\_queue,)).start()  
 multiprocessing.Process(target=handle\_upload\_file, args=(upload\_queue, files\_obj, port, list\_queue, path\_of\_file,)).start()  
 *# send to the client the port* response = serverProtocol.serverProtocol.Response\_for\_upload\_request(path\_of\_file, port)  
 general\_comm.send(response, ip)  
  
  
**def** \_after\_upload(list\_queue):  
 *"""  
 send update to the client and update the list of open files* **:param** *list\_queue: the queue connect to the upload process  
 """* boolean = **True** file\_name, ip, path, len\_data = list\_queue.get()  
 **if** len\_data == -2:  
 boolean = **False  
 elif** len\_data != -1:  
 open\_files[file\_name] = [len\_data, 0]  
 response = serverProtocol.serverProtocol.Response\_for\_upload(boolean, path)  
 general\_comm.send(response, ip)  
  
  
**def** handle\_disconnect\_nitur(params):  
 *"""  
 if client disconnect delete all the files that he had and send to everyone* **:param** *params: list of params* **:return***:  
 """* ip = params[1]  
 *# all the names that the ip had* list\_of\_names = files\_obj.handdle\_disconnect(ip)  
 **for** name **in** list\_of\_names:  
 **if** name **in** open\_files:  
 open\_files[name][1] = max(0, open\_files[name][1] - 1)  
 string\_of\_open\_files = serverProtocol.serverProtocol.Create\_string\_of\_list(open\_files)  
 general\_comm.sendall(string\_of\_open\_files)  
  
  
**def** create\_open\_files():  
 *"""  
 create the list of file with all the files from the torrent file  
  
 """* files\_name = os.listdir(settingSer.PATH\_OF\_TORRENTS\_FOLDER)  
 **for** name **in** files\_name:  
 name = name[:-5]  
 torrent = files\_obj.get\_torrent\_file(name, string=**False**)  
 open\_files[name] = [torrent[**"len of file"**], 0]  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# all the opcodes and funcs* general\_commands = {**"01"**: send\_torrent, **"02"**: create\_uplaod\_socket, **"03"**: send\_list\_of\_files}  
 *# all the opcodes and funcs* nitur\_commands = {**"-1"**: handle\_disconnect\_nitur, **"01"**: file\_added, **"02"**: file\_deleted, **"03"**: file\_changed, **"05"**: file\_name\_changed}  
  
 *# the files object* files\_obj = ServerFiles.Server\_files(settingSer.PATH\_OF\_TORRENTS\_FOLDER)  
  
 open\_files = {}  
 create\_open\_files()  
  
 unused\_ports = (x **for** x **in** range(2000, 2500))  
  
 general\_queue = multiprocessing.Queue()  
 nitur\_queue = queue.Queue()  
  
 *# the general comm* general\_comm = ServerComm.ServerComm(settingSer.GENERAL\_PORT, general\_queue, 6)  
 *# the nitur comm* nitur\_comm = ServerComm.ServerComm(settingSer.NITUR\_PORT, nitur\_queue, 2)  
  
 *# start the handle* threading.Thread(target=handle\_general\_msgs, args=(general\_queue,)).start()  
 threading.Thread(target=handle\_nitur\_msgs, args=(nitur\_queue,)).start()

**from** serverCode **import** DB  
**from** allcode **import** ServerComm  
  
  
**def** handle\_upload\_file(upload\_queue, files\_obj, port, update\_list\_queue, path\_of\_file):  
 *"""  
 open a server and recv a file from the client* **:param** *upload\_queue: the queue from the comm* **:param** *files\_obj: files object* **:param** *port: port of server* **:param** *update\_list\_queue: queue to the logic  
 -1 same file already exist, -2 same name already exists  
 """* upload\_comm = ServerComm.ServerComm(port, upload\_queue, 8)  
 torrents\_db = DB.DBClass()  
 ip, message = upload\_queue.get()  
 file\_name, data = message[1], message[2]  
 **if not** torrents\_db.have\_torrent(file\_name):  
 *# create the torrent file* files\_obj.create\_torrent\_file(data, file\_name)  
 *# add the torrent file to the db* torrents\_db.add\_torrent(file\_name)  
 *# add the file name to the list of file* update\_list\_queue.put((file\_name, ip, path\_of\_file, len(data)))  
 **else**:  
 temp = **f"temp{**port**}"** files\_obj.create\_torrent\_file(data, temp)  
  
 *# check if the file is the same as the origin file, create a torrent and compare if they are the same  
 # (without the same name and open ip header* temp\_torrent = files\_obj.get\_torrent\_file(temp, **False**)  
 temp\_torrent = list(temp\_torrent.values())[2:]  
  
 files\_obj.delete\_temp(temp)  
  
 origin\_torrent = files\_obj.get\_torrent\_file(file\_name, **False**)  
 origin\_torrent = list(origin\_torrent.values())[2:]  
  
 **if** temp\_torrent == origin\_torrent:  
 files\_obj.add\_ip\_to\_torrent(file\_name, ip)  
 update\_list\_queue.put((file\_name, ip, path\_of\_file, -1))  
 **else**:  
 update\_list\_queue.put((file\_name, ip, path\_of\_file, -2))  
 torrents\_db.closeDb()  
 upload\_comm.close\_socket()

**from** allcode **import** Encryption\_Decryption  
**import** math  
**import** json  
**import** os  
**import** ctypes  
**from** allcode **import** settingSer  
  
  
**class** Server\_files(object):  
 **def** \_\_init\_\_(self, torrent\_files\_path):  
 *"""* **:param** *torrent\_files\_path: the path of the torrent  
 """* self.torrent\_files\_path = torrent\_files\_path  
 *# create the dir* **if not** os.path.exists(torrent\_files\_path):  
 os.mkdir(torrent\_files\_path)  
 attrs = ctypes.windll.kernel32.GetFileAttributesW(torrent\_files\_path)  
 *# Add the 'hidden' attribute* ctypes.windll.kernel32.SetFileAttributesW(torrent\_files\_path, attrs | 2)  
  
 **def** create\_torrent\_file(self, data, filename):  
 *"""  
 create the torrent file* **:param** *data: the data of the file* **:param** *filename: the name of the file  
 """* list\_of\_hash = []  
 **for** i **in** range(math.ceil(len(data)/settingSer.len\_of\_part\_of\_file)):  
 data\_part = data[i\*settingSer.len\_of\_part\_of\_file:(i+1)\*settingSer.len\_of\_part\_of\_file]  
 list\_of\_hash.append(str(Encryption\_Decryption.AES\_encryption.hash(data\_part)))  
 full\_hash = str(Encryption\_Decryption.AES\_encryption.hash(data))  
 torrent\_file = self.\_build\_torrent\_file\  
 (data, filename, full\_hash, list\_of\_hash, settingSer.len\_of\_part\_of\_file)  
 **with** open(**f"{**self.torrent\_files\_path**}\\{**filename**}.json"**, **'w'**) **as** f:  
 json.dump(torrent\_file, f)  
  
 **def** \_build\_torrent\_file(self, file\_data, file\_name, full\_hash, parts\_hash, len\_of\_part):  
 *"""* **:param** *file\_data: the data* **:param** *file\_name: the name* **:param** *full\_hash: the full hash of the file* **:param** *parts\_hash: list of the parts hash* **:return***:  
 """* json\_file = {**"file name"**: file\_name, **"open ip"**: [],  
 **"len of file"**: len(file\_data), **"number of pieces"**: len(parts\_hash),  
 **"hash of pieces"**: parts\_hash, **"full hash"**: full\_hash, **"part len"**: len\_of\_part}  
 **return** json\_file  
  
 **def** add\_ip\_to\_torrent(self, file\_name, ip):  
 *"""  
 add ip to the list of open ips* **:param** *file\_name: the name* **:param** *ip: the ip  
 """* **with** open(**f"{**self.torrent\_files\_path**}\\{**file\_name**}.json"**, **"r"**) **as** f:  
 data = json.load(f)  
  
 list\_of\_ip = data[**"open ip"**]  
 **if** ip **not in** list\_of\_ip:  
 list\_of\_ip.append(ip)  
 data[**"open ip"**] = list\_of\_ip  
  
 **with** open(**f"{**self.torrent\_files\_path**}\\{**file\_name**}.json"**, **"w"**) **as** f:  
 json.dump(data, f)  
 **return  
  
 def** get\_torrent\_file(self, file\_name, string=**True**):  
 *"""* **:param** *file\_name: the name* **:param** *string: return as a string* **:return***: the torrent  
 """* data = **None  
 if** os.path.exists(**f"{**self.torrent\_files\_path**}\\{**file\_name**}.json"**):  
 **with** open(**f"{**self.torrent\_files\_path**}\\{**file\_name**}.json"**, **"r"**) **as** f:  
 data = json.load(f)  
 **if** string:  
 data = json.dumps(data)  
 **return** data  
  
 **def** delete\_ip\_from\_torrent(self, file\_name, ip):  
 *"""  
 remove the ip from the open ip list* **:param** *file\_name: the name of the file* **:param** *ip: the ip  
 """* boolean = **False  
 with** open(**f"{**self.torrent\_files\_path**}\\{**file\_name**}.json"**, **"r"**) **as** f:  
 data = json.load(f)  
  
 list\_of\_ip = data[**"open ip"**]  
 **if** ip **in** list\_of\_ip:  
 list\_of\_ip.remove(ip)  
 boolean = **True** data[**"open ip"**] = list\_of\_ip  
  
 **with** open(**f"{**self.torrent\_files\_path**}\\{**file\_name**}.json"**, **"w"**) **as** f:  
 json.dump(data, f)  
 **return** boolean  
  
 **def** handdle\_disconnect(self, ip):  
 *"""  
 remove the ip of every torrent file* **:param** *ip: the ip of the client* **:return***: list of the file that have been deleted  
 """* list\_of\_deleted\_ips = []  
 files\_name = os.listdir(self.torrent\_files\_path)  
 **for** name **in** files\_name:  
 boolean = self.delete\_ip\_from\_torrent(name[:-5], ip)  
 **if** boolean:  
 list\_of\_deleted\_ips.append(name[:-5])  
  
 **return** list\_of\_deleted\_ips  
  
 **def** delete\_temp(self, file\_name):  
 *"""  
 delete the temp torrent* **:param** *file\_name: file name* **:return***:  
 """* **if** os.path.exists(**f"{**self.torrent\_files\_path**}\\{**file\_name**}"**):  
 os.remove(**f"{**self.torrent\_files\_path**}\\{**file\_name**}"**)

**class** serverProtocol(object):  
 @staticmethod  
 **def** unpack(data):  
 *"""* **:param** *data: the data from the comm* **:return***:opcdoe and list of params  
 """* params = data.split(**"$%$"**)  
 opcode = params[0]  
 params.remove(params[0])  
 **return** opcode, params  
  
 @staticmethod  
 **def** unpack\_file(data):  
 *"""* **:param** *data:the data of the comm* **:return***: opcode and list of params  
 """* params = data.split(**"$%$"**)  
 opcode = params[1]  
 params.remove(params[1])  
 **return** opcode, params  
  
 @staticmethod  
 **def** build\_part\_of\_key(B):  
 *"""* **:param** *B: part of the key* **:return***: string using format  
 """* **return f"00{**B**}"** @staticmethod  
 **def** Response\_for\_torrent\_request(file\_name):  
 *"""* **:param** *file\_name:the name of the file* **:return***:string using format  
 """* **return f"01$%$0$%${**file\_name**}$%$"** @staticmethod  
 **def** Response\_for\_upload\_request(path\_of\_file, port):  
 *"""* **:param** *path\_of\_file: the path of the file* **:param** *port: the port* **:return***: string using format  
 """* **return f"02$%${**port**}$%${**path\_of\_file**}"** @staticmethod  
 **def** Response\_for\_upload(bol, path):  
 *"""* **:param** *bol: fail or pass* **:return***: string protocol  
 """* code = **"222"  
 if** bol:  
 code = **"111"  
 return f"04$%${**code**}$%${**path**}"** @staticmethod  
 **def** Delete\_file\_from\_folder(file\_name):  
 *"""* **:param** *file\_name: the file name* **:return***: string using format  
 """* **return f"03$%${**file\_name**}"** @staticmethod  
 **def** Create\_string\_of\_list(list\_of\_open\_file):  
 *"""* **:param** *list\_of\_open\_file: list of all the open clients* **:return***: string with the info of each file in the format  
 """* message = **"03"  
 for** file **in** list\_of\_open\_file:  
 **if** list\_of\_open\_file[file][1] != 0:  
 message += **f"$%${**file**},{**list\_of\_open\_file[file][0]**},{**list\_of\_open\_file[file][1]**}"  
 return** message

**import** sqlite3  
  
  
**class** DBClass:  
 **def** \_\_init\_\_(self):  
 *"""  
 set the curr and the conns and the names  
 """* self.dbName = **"NexusDB.sql"** self.tblName = **"torrnetTBL"** self.conn = **None** self.curr = **None** self.\_buildDB()  
  
 **def** \_buildDB(self):  
 *"""  
 create DB if DB dose not exists yet* **:return***: None  
 """* self.conn = sqlite3.connect(self.dbName)  
 self.curr = self.conn.cursor()  
 sql = **f"CREATE TABLE IF NOT EXISTS {**self.tblName**}(names VARCHAR(10) PRIMARY KEY)"** self.curr.execute(sql)  
 self.conn.commit()  
  
 **def** have\_torrent(self, torrent\_name):  
 *"""  
 check if user name in the table* **:param** *torrent\_name:* **:return***: none or data  
 """* sql = **f"SELECT \* FROM {**self.tblName**} WHERE names = ?"** self.curr.execute(sql, (torrent\_name,))  
 **return** self.curr.fetchone() **is not None  
  
 def** add\_torrent(self, torrent\_name):  
 *"""* **:param** *torrent\_name:* **:return***:  
 """* **if not** self.have\_torrent(torrent\_name):  
 sql = **f"INSERT INTO {**self.tblName**} VALUES (?)"** self.curr.execute(sql, (torrent\_name,))  
 self.conn.commit()  
  
 **def** deleteDb(self):  
 *"""  
 delete the db* **:return***:  
 """* sql = **f"DROP TABLE {**self.tblName**}"** self.curr.execute(sql)  
 self.conn.commit()  
  
 **def** closeDb(self):  
 *"""  
 close the db* **:return***:  
 """* self.curr.close()  
 self.conn.close()

**from** clientCode **import** clientProtocol  
**from** clientCode **import** ClientComm  
**from** clientCode **import** ClientFiles  
**from** allcode **import** ServerComm  
**from** clientCode **import** monitoring  
**import** queue  
**from** allcode **import** settingCli  
**import** threading  
  
  
**def** handle\_p2p\_msgs(p2p\_server\_queue):  
 *"""  
 calls the func by the opcode* **:param** *p2p\_server\_queue: the queue of the p2p messages  
 """* **while True**:  
 ip, message = p2p\_server\_queue.get()  
 opcode, params = clientProtocol.clientProtocol.unpack(message)  
 params.append(ip)  
 p2p\_server\_commands[opcode](params)  
  
  
**def** send\_part\_of\_file(params):  
 *"""  
 send the part of the file* **:param** *params: the params  
 """* file\_part, file\_name, len\_part, ip = int(params[0]), params[1], int(params[2]), params[3]  
 data\_of\_part = ClientFiles.client\_files. \  
 get\_part\_of\_file(**f"{**settingCli.NITUR\_FOLDER**}\\{**file\_name**}"**, file\_part)  
 header = clientProtocol.clientProtocol.send\_file\_part(file\_name, file\_part)  
 p2p\_server.send\_file(data\_of\_part, header, ip)  
  
  
**def** handle\_nitur\_comm\_msgs(nitur\_comm\_queue):  
 *"""  
 calls the func by the opcode* **:param** *nitur\_comm\_queue: the queue of the messages from the server nitur comm  
 """* **while True**:  
 message = nitur\_comm\_queue.get()  
 opcode, file\_name = clientProtocol.clientProtocol.unpack(message)  
 **if** opcode == **"03"**:  
 ClientFiles.client\_files.delete\_file(**fr"{**settingCli.NITUR\_FOLDER**}\\{**file\_name[0]**}"**)  
  
  
**def** handle\_nitur\_msgs(nitur\_queue):  
 *"""  
 calls the func by the opcode* **:param** *nitur\_queue: the queue of the nitur messages  
 """* **while True**:  
 opcode, file\_name = nitur\_queue.get()  
 nitur\_commands[opcode](file\_name)  
  
  
**def** delete\_file(file\_name):  
 *"""  
 send to the server that a file been deleted from the nitur* **:param** *file\_name: the name  
 """* update\_in\_nitur = clientProtocol.clientProtocol.removed\_file\_nitur(file\_name)  
 nitur\_comm.send(update\_in\_nitur)  
  
  
**def** add\_file(file\_name):  
 *"""  
 send to the server that a file been added to the nitur* **:param** *file\_name: the file name  
 """* update\_in\_nitur = clientProtocol.clientProtocol.added\_file\_nitur(file\_name)  
 nitur\_comm.send(update\_in\_nitur)  
  
  
**def** change\_file(file\_name):  
 *"""  
 send to the server that a file been changed in the nitur* **:param** *file\_name: the name  
 """* update\_in\_nitur = clientProtocol.clientProtocol.changed\_file\_nitur(file\_name)  
 nitur\_comm.send(update\_in\_nitur)  
  
  
**def** change\_file\_name(file\_name):  
 *"""  
 send to the server that a file name been changed in the nitur* **:param** *file\_name: the name  
 """* update\_in\_nitur = clientProtocol.clientProtocol.changed\_file\_name\_new\_nitur(file\_name)  
 nitur\_comm.send(update\_in\_nitur)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 *# create the nitur object* nitur\_queue = queue.Queue()  
 nitur = monitoring.monitoring(settingCli.NITUR\_FOLDER, nitur\_queue)  
  
 *# create server for a p2p comm* p2p\_server\_queue = queue.Queue()  
 p2p\_server = ServerComm.ServerComm(settingCli.P2P\_PORT, p2p\_server\_queue, 4)  
  
 *# create client comm for the nitur messages* nitur\_comm\_queue = queue.Queue()  
 nitur\_comm = ClientComm.Clientcomm(settingCli.SERVER\_IP, nitur\_comm\_queue, settingCli.NITUR\_PORT, 2)  
  
 *# all the nitur commands and funcs* nitur\_commands = {**"01"**: add\_file, **"02"**: delete\_file, **"03"**: change\_file, **"05"**: change\_file\_name}  
  
 p2p\_server\_commands = {**"01"**: send\_part\_of\_file}  
  
 *# start as a thread the handles* threading.Thread(target=handle\_nitur\_comm\_msgs, args=(nitur\_comm\_queue,)).start()  
 threading.Thread(target=handle\_nitur\_msgs, args=(nitur\_queue,)).start()  
 threading.Thread(target=handle\_p2p\_msgs, args=(p2p\_server\_queue,)).start()

**from** clientCode **import** clientProtocol  
**from** clientCode **import** ClientFiles  
**from** clientCode **import** ClientComm  
**from** clientCode.p2pProcess **import** \_build\_com  
**from** allcode **import** Encryption\_Decryption  
**import** queue  
**import** threading  
**import** os  
**from** allcode **import** settingCli  
**import** json  
**import** multiprocessing  
**import** wx  
**from** clientCode **import** graphics  
**from** pubsub **import** pub  
**import** time  
  
  
**def** handle\_general\_msgs(general\_queue):  
 *"""  
 call the func by the opcode* **:param** *general\_queue: the queue of the general comm  
 """* **while True**:  
 message = general\_queue.get()  
 **if** type(message) == tuple:  
 p2p\_download(message)  
 **else**:  
 opcode, params = clientProtocol.clientProtocol.unpack(message)  
 general\_commands[opcode](params)  
  
  
**def** handle\_message\_from\_gui(gui\_queue):  
 *"""  
 send to the server the command the user did* **:param** *gui\_queue: the queue connect the qui to the logic  
 """* **global** abort  
 **while True**:  
 order, path = gui\_queue.get()  
 **if** order == **"download"**:  
 message = clientProtocol.clientProtocol.request\_torrent\_file(path)  
 **elif** order == **"upload"**:  
 message = clientProtocol.clientProtocol.request\_upload(path)  
 **elif** order == **"abort"**:  
 abort = **True** *# close the loop of the download* **continue  
 else**:  
 **continue** general\_comm.send(message)  
  
  
**def** create\_socket\_upload(params):  
 *"""  
 create a comm and upload the file* **:param** *params:* **:return***:  
 """* port, path\_of\_file = int(params[0]), params[1]  
 **if** port != -1:  
 upload\_queue = queue.Queue()  
 file\_name = os.path.basename(path\_of\_file)  
 *# create comm* upload\_comm = ClientComm.Clientcomm(server\_ip, upload\_queue, port, 8)  
 data = ClientFiles.client\_files.get\_part\_of\_file(path\_of\_file, -1)  
 header = clientProtocol.clientProtocol.upload\_file(file\_name)  
 *# send the data* upload\_comm.send\_file(data, header)  
  
  
**def** p2p\_download(params):  
 *"""  
 download the file using p2p connection* **:param** *params:list of the params  
 """* json\_string = params[3]  
 list\_of\_processes = []  
 torrnet\_dict = json.loads(json\_string)  
 file\_name = torrnet\_dict[**"file name"**]  
 len\_of\_part = int(torrnet\_dict[**"part len"**])  
 file\_queue = multiprocessing.Queue()  
 comms = {}  
 open\_ips\_dict = torrnet\_dict[**'open ip'**]  
 number\_of\_pieces = torrnet\_dict[**'number of pieces'**]  
 *# create list of pieces* list\_of\_pieces = [0 **for** \_ **in** range(number\_of\_pieces)]  
 wx.CallAfter(pub.sendMessage, **"start progress"**, total\_parts=number\_of\_pieces,  
 name\_of\_file=file\_name)  
  
 *# start a thread to get the parts from the queue* threading.Thread(target=\_get\_parts\_from\_queue, args=(comms, torrnet\_dict[**'hash of pieces'**], torrnet\_dict[**'full hash'**], file\_queue, file\_name, list\_of\_pieces, len\_of\_part,)).start()  
  
 *# for each ip create comm* **for** ip **in** open\_ips\_dict:  
 ip\_queue = multiprocessing.Queue()  
 comms[ip] = [ip\_queue, 0]  
 *# start a process for each ip* p = multiprocessing.Process(target=\_build\_com, args=(ip\_queue, file\_queue, ip, file\_name,len\_of\_part))  
 p.start()  
 list\_of\_processes.append(p)  
 *# get the index of the next part* index = \_find\_first(list\_of\_pieces, -1, -1)  
 ip\_queue.put(index)  
  
  
**def** \_get\_parts\_from\_queue(comms, list\_of\_hash, full\_data\_hash, file\_queue, file\_name, list\_of\_pieces, len\_of\_part):  
 *"""  
  
 get the parts of the file from the queue check the hash and update the graphics* **:param** *comms: list of comm* **:param** *list\_of\_hash: list of parts hash* **:param** *full\_data\_hash: the full hash of the data* **:param** *file\_queue:the queue connect to the every process* **:param** *file\_name: the name of the file* **:param** *list\_of\_pieces: the list of each piece status  
 """* **global** abort  
 build\_file = queue.Queue()  
 path = **f"{**settingCli.PATH\_TO\_SAVE\_FILES**}**\**{**file\_name**}"** first\_time = time.time()  
 *# thread that build the file* threading.Thread(target=\_build\_file, args=(build\_file, path, len\_of\_part)).start()  
 **while not** abort:  
 ip, file\_name, number\_of\_part, data = file\_queue.get()  
 *# not a time out* **if** data != -1:  
 hash\_part = Encryption\_Decryption.AES\_encryption.hash(data)  
 *# check hash  
 # bad hash* **if** list\_of\_hash[number\_of\_part] != str(hash\_part):  
 *# disconnect* comms[ip][0].put(-1)  
 **del** comms[ip]  
 *# if last comm* **if** len(comms) == 0:  
 print(**"Fail - hash are not the same "**)  
 **break** *# good hash* **else**:  
 *# add the part to the file* build\_file.put((data, number\_of\_part))  
 *# let the graphics know* wx.CallAfter(pub.sendMessage, **"new part"**)  
  
 *# not the first part anymore  
 # get the index of the next part to download* index = \_find\_first(list\_of\_pieces, -1, number\_of\_part)  
 *# all the part are done* **if** index == -1:  
 *# end the func that build the file* **break** *# ask for the next part* comms[ip][0].put(index)  
 *# couldn't find part* **else**:  
 *# after 3 try's* **if** comms[ip][1] == 2:  
 *# close the comm* comms[ip][0].put(-1)  
 **del** comms[ip]  
 **if** len(comms) == 0:  
 print(**"Fail - to many try's, closed socket"**)  
 **break  
 else**:  
 *# add a try* comms[ip][1] += 1  
 index = \_find\_first(list\_of\_pieces, number\_of\_part)  
 *# could find last part* **if** index == -2:  
 comms[ip][0].put(-1)  
 **del** comms[ip]  
 **if** len(comms) == 0:  
 print(**"Fail - last part wasn't found"**)  
 *# send to gui* **break  
 else**:  
 comms[ip][0].put(index)  
 print(**f"the time to download is {**time.time() - first\_time**}"**)  
  
 data = ClientFiles.client\_files.get\_part\_of\_file(path, -1)  
  
 full\_hash = Encryption\_Decryption.AES\_encryption.hash(data)  
 *# check the full hash* **if** full\_data\_hash == str(full\_hash):  
 *# confirm that the full hash is good* wx.CallAfter(pub.sendMessage, **"new part"**)  
  
 ClientFiles.client\_files.save\_file(settingCli.NITUR\_FOLDER, file\_name, data)  
  
 wx.CallAfter(pub.sendMessage, **"close progress"**)  
 *# close all the sockets* **for** ip **in** comms:  
 comms[ip][0].put(-1)  
 abort = **False** build\_file.put((0, -1))  
  
  
**def** \_build\_file(build\_queue, path, len\_of\_part):  
 *"""P  
 open the file, build the file with data from the queue and close the file* **:param** *build\_queue: the queue* **:param** *path: the path  
 """* f = open(path, **'wb'**)  
 **while True**:  
 data, number\_of\_part = build\_queue.get()  
 *# -1 = last part* **if** number\_of\_part **is** -1:  
 **break** f.seek(number\_of\_part\*len\_of\_part)  
 f.write(data)  
 f.close()  
  
  
**def** \_find\_first(list\_of\_pieces, cant\_recv, number\_found=-1):  
 *"""* **:param** *list\_of\_pieces: the list of the status* **:param** *cant\_recv: if part fail, dont ask for him again* **:param** *number\_found: the number of part that was found* **:return***:  
 """  
 # if a number was found* **if** number\_found != -1:  
 *# change the status of the part* list\_of\_pieces[number\_found] = -1  
 *# if all the parts found return -1* **if** list\_of\_pieces.count(-1) == len(list\_of\_pieces):  
 index = -1  
 **else**:  
 **if** cant\_recv != -1:  
 *# if a part wasn't found change the status* list\_of\_pieces[cant\_recv] -= 1  
 *# find new part to ask* number\_list = [n **for** n **in** list\_of\_pieces **if** n >= 0 **and** n != cant\_recv]  
 *# if no other part was found* **if** len(number\_list) == 0:  
 *# delete the comm* index = -2  
 **else**:  
 *# ask for the part with the min number of working clients on him* min\_number = min(number\_list)  
 index = list\_of\_pieces.index(min\_number)  
 *# change the status* list\_of\_pieces[index] += 1  
 **return** index  
  
  
**def** create\_list\_of\_files(params):  
 *"""  
 call a func in the graphics to update the list of file* **:param** *params: list of files  
 """* params = [params]  
 list\_of\_files = [[item.split(**','**) **for** item **in** sublist] **for** sublist **in** params]  
 wx.CallAfter(pub.sendMessage, **"new list"**, new\_file\_list=list\_of\_files[0])  
  
  
**def** get\_response\_server(params):  
 *"""  
 get the response from the server after upload and save the file if dosen't exists* **:param** *params:* **:return***:  
 """* response, path = params[0], params[1]  
 **if** response == **"111"**:  
 data = ClientFiles.client\_files.get\_part\_of\_file(path, -1)  
 file\_name = os.path.basename(path)  
 ClientFiles.client\_files.save\_file(settingCli.NITUR\_FOLDER, file\_name, data)  
 wx.CallAfter(pub.sendMessage, **"after upload"**)  
 **else**:  
 *# if file already exists* wx.CallAfter(pub.sendMessage, **"file exists"**)  
  
  
**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 server\_ip = settingCli.SERVER\_IP  
 *# all the general opcodes and funcs* general\_commands = {**"01"**: p2p\_download, **"02"**: create\_socket\_upload, **"03"**: create\_list\_of\_files  
 , **"04"**: get\_response\_server}  
  
 general\_queue = queue.Queue()  
 gui\_queue = queue.Queue()  
 abort = **False** *# create the general comm* general\_comm = ClientComm.Clientcomm(server\_ip, general\_queue, 1500, 6)  
 *# start the handles* threading.Thread(target=handle\_general\_msgs, args=(general\_queue, )).start()  
 threading.Thread(target=handle\_message\_from\_gui, args=(gui\_queue,)).start()  
 *# start the main thread of the graphics* app = wx.App()  
 frame = graphics.MyFrame(**None**, **"nexus"**, gui\_queue, logo\_path=settingCli.IMAGES\_PATH)  
 app.MainLoop()

**from** clientCode **import** clientProtocol  
**from** clientCode **import** ClientComm  
**from** allcode **import** settingCli  
  
  
**def** \_build\_com(main\_queue, file\_queue, ip, file\_name, len\_of\_part):  
 *"""* **:param** *main\_queue: the queue connect to the logic* **:param** *file\_queue: the queue that recv the files* **:param** *ip: the ip of the server* **:param** *file\_name:the name of the file  
 """* comm = ClientComm.Clientcomm(ip, file\_queue, settingCli.P2P\_PORT, 4)  
 **while True**:  
 index = main\_queue.get()  
 *# close the socket* **if** index == -1:  
 comm.close\_socket()  
 **else**:  
 request\_part = clientProtocol.clientProtocol.request\_part\_file(file\_name, index, len\_of\_part)  
 comm.send(request\_part)

**import** win32file  
**import** win32con  
**import** win32event  
**import** threading  
**import** os  
**import** ctypes  
  
  
**class** monitoring(object):  
 **def** \_\_init\_\_(self, path\_to\_monitor, queue):  
 *"""* **:param** *path\_to\_monitor: the path of the hidden file* **:param** *queue: the queue to put all the changes of the folder  
 """* self.path\_to\_monitor = path\_to\_monitor  
 self.msgs\_queue = queue  
 self.list\_of\_new\_files = []  
 *# if there is no dir exists create one* **if not** os.path.exists(path\_to\_monitor):  
 os.mkdir(path\_to\_monitor)  
 attrs = os.stat(self.path\_to\_monitor).st\_file\_attributes  
  
 *# Check if the 'hidden' attribute is set* **if not** attrs & 2 == 2:  
 *# Get the file attributes* attrs = ctypes.windll.kernel32.GetFileAttributesW(self.path\_to\_monitor)  
 *# Add the 'hidden' attribute* ctypes.windll.kernel32.SetFileAttributesW(self.path\_to\_monitor, attrs | 2)  
  
 *# send all the existing files to the server* self.\_send\_existing\_files()  
 *# start the monitoring thread* threading.Thread(target=self.\_monitoring\_folder).start()  
 *# make the first change so start the monitoring* self.first\_change()  
  
 **def** first\_change(self):  
 *"""  
 make the first change  
 """* path = **f"{**self.path\_to\_monitor**}\\a.txt"  
 with** open(path, **'w'**) **as** f:  
 f.write(**"imri"**)  
 **if** os.path.exists(**fr"{**path**}"**):  
 os.remove(**fr"{**path**}"**)  
  
 **def** \_send\_existing\_files(self):  
 *"""  
 send all the existing files to the serve  
 """* list\_of\_file = os.listdir(self.path\_to\_monitor)  
 **for** file\_name **in** list\_of\_file:  
 self.msgs\_queue.put((**"01"**, file\_name))  
  
 **def** \_monitoring\_folder(self):  
 *'''  
 monitor a directory and record the changes in a file and on the screen* **:return***:  
 '''  
 # set up the notification handle* change\_handle = win32file.FindFirstChangeNotification(  
 self.path\_to\_monitor,  
 0,  
 win32con.FILE\_NOTIFY\_CHANGE\_FILE\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_DIR\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_SIZE  
  
 )  
 **try**:  
 **while True**:  
 result = win32event.WaitForSingleObject(change\_handle, win32event.INFINITE)  
 *# check if there are any changes* **if** result == win32event.WAIT\_OBJECT\_0:  
 **for** action, filename **in** win32file.ReadDirectoryChangesW(change\_handle, 1024, **True**,  
 win32con.FILE\_NOTIFY\_CHANGE\_FILE\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_DIR\_NAME | win32con.FILE\_NOTIFY\_CHANGE\_LAST\_WRITE):  
 *# add file* **if** action == 1:  
 self.list\_of\_new\_files.append(filename)  
 *# delete file* **elif** action == 2:  
 self.msgs\_queue.put((**"02"**, filename))  
 *# change file* **elif** action == 3:  
 **if** filename **in** self.list\_of\_new\_files:  
 self.list\_of\_new\_files.remove(filename)  
 self.msgs\_queue.put((**"01"**, filename))  
 **else**:  
 self.msgs\_queue.put((**"03"**, filename))  
 *# change file* **elif** action == 4:  
 self.msgs\_queue.put((**"03"**, filename))  
 *# change name file* **elif** action == 5:  
 self.msgs\_queue.put((**"05"**, filename))  
 **except** Exception **as** e:  
 win32file.FindCloseChangeNotification(change\_handle)

**import** wx  
**import** wx.adv  
**from** humanize **import** naturaldelta, naturalsize  
**import** os  
**from** pubsub **import** pub  
**from** allcode **import** settingCli  
**import** shutil  
  
  
**class** MyFrame(wx.Frame):  
 **def** \_\_init\_\_(self, parent, title, queue, logo\_path):  
 *"""  
 create the frame and all the buttons* **:param** *parent: None* **:param** *title: the title* **:param** *queue: the queue to put the messages* **:param** *logo\_path: the path to the logo  
 """* super(MyFrame, self).\_\_init\_\_(parent, title=title, size=(680, 350))  
  
 *# don't allow to change screen size* self.SetMaxSize((680, 350))  
 self.SetMinSize((680, 350))  
  
 self.panel = wx.Panel(self)  
 self.panel.SetBackgroundColour(**f"#001f3f"**)  
 self.SetIcon(wx.Icon(**f"{**logo\_path**}icon.png"**, wx.BITMAP\_TYPE\_PNG))  
 self.queue = queue  
 *# List of lists: [file\_name, size, number of clients]* self.file\_list = []  
 self.progress\_dialog = **None** self.total\_parts = 0  
 self.total\_parts\_download = 0  
 self.name\_of\_file = **""** *# help\_icon = wx.ArtProvider.GetBitmap(wx.ART\_INFORMATION, wx.ART\_OTHER, (35, 35))* help\_icon = wx.Bitmap(**fr"T:\public\יב\imri\nexus-main\info.png"**)  
 icon\_button = wx.StaticBitmap(self.panel, bitmap=help\_icon, pos=(624, 16))  
 icon\_button.Bind(wx.EVT\_LEFT\_DOWN, self.show\_info\_dialog)  
  
 *# upload button* self.upload\_button = wx.Button(self.panel, label=**"Upload"**, pos=(30, 220), size=(100, 60))  
 font = wx.Font(11, wx.DEFAULT, wx.NORMAL, wx.NORMAL, faceName=**"Arial Rounded MT Bold"**)  
 self.upload\_button.SetFont(font)  
  
 *# download button* self.download\_button = wx.Button(self.panel, label=**"Download"**, pos=(160, 220), size=(100, 60))  
 font = wx.Font(11, wx.DEFAULT, wx.NORMAL, wx.NORMAL, faceName=**"Arial Rounded MT Bold"**)  
 self.download\_button.SetFont(font)  
 *# the table of the files* self.file\_list\_ctrl = wx.ListCtrl(self.panel, style=wx.LC\_REPORT | wx.BORDER\_SUNKEN, pos=(35, 15),  
 size=(585, 170))  
 self.file\_list\_ctrl.SetBackgroundColour(wx.Colour(255, 255, 255))  
 self.file\_list\_ctrl.InsertColumn(0, **'File Name'**, width=140)  
 self.file\_list\_ctrl.InsertColumn(1, **'Size'**, width=140)  
 self.file\_list\_ctrl.InsertColumn(2, **'Clients'**, width=130)  
 self.file\_list\_ctrl.InsertColumn(3, **'estimated time'**, width=165)  
  
 *# change the colors to the button* self.download\_button.SetBackgroundColour(**F"#36bf81"**)  
 self.download\_button.SetForegroundColour(wx.Colour(255, 255, 255))  
  
 *# change the colors to the button* self.upload\_button.SetBackgroundColour(**F"#36bf81"**)  
 self.upload\_button.SetForegroundColour(wx.Colour(255, 255, 255))  
  
 *# set the func to call when the button is click* self.Bind(wx.EVT\_BUTTON, self.on\_upload, self.upload\_button)  
 self.Bind(wx.EVT\_BUTTON, self.on\_download, self.download\_button)  
  
 *# put the logo* image\_path = **f"{**logo\_path**}mylogo.png"** image = wx.Image(image\_path, wx.BITMAP\_TYPE\_ANY)  
 image = image.Scale(380, 75, wx.IMAGE\_QUALITY\_HIGH)  
 wx.StaticBitmap(self.panel, bitmap=wx.Bitmap(image), pos=(280, 220))  
  
 self.Show()  
  
 *# bind the func to the strings* pub.subscribe(self.update\_file\_list, **"new list"**)  
 pub.subscribe(self.open\_progress, **"start progress"**)  
 pub.subscribe(self.close\_progress, **"close progress"**)  
 pub.subscribe(self.show\_download\_progress, **"new part"**)  
 pub.subscribe(self.file\_exists\_error, **"file exists"**)  
 pub.subscribe(self.after\_upload, **"after upload"**)  
  
 **def** after\_upload(self):  
 *"""  
 shows that the upload was good  
 """  
 # Show a message box when the upload is complete* wx.MessageBox(**f"Upload complete!"**, **"Success"**, wx.OK | wx.ICON\_INFORMATION,  
 self)  
 self.upload\_button.Enable(**True**)  
  
 **def** show\_info\_dialog(self, event):  
 *"""  
 shows info about the project  
 """* info = wx.adv.AboutDialogInfo()  
 info.SetName(**"imri system"**)  
 info.SetDescription(**"this system allow you to download files faster and safer"**)  
 info.SetCopyright(**"(C) 2024-2030"**)  
 info.AddDeveloper(**"Imri Hilel"**)  
 info.SetVersion(**"2.0"**)  
 info.SetName(**"nexus"**)  
 info.SetLicence(**"Microsoft \n Google \n NVIDIA"**)  
 wx.adv.AboutBox(info)  
  
 **def** file\_exists\_error(self):  
 *"""  
 show error when file already exists  
 """* wx.MessageBox(**"file with that name already exists in the system"**,  
 **"Error"**,  
 wx.OK | wx.ICON\_WARNING)  
 self.upload\_button.Enable(**True**)  
  
 **def** on\_upload(self, event):  
 *"""  
 shows the file explorer and let the user choose a file  
 """* wildcard = **"All files (\*.\*)|\*.\*"** dialog = wx.FileDialog(self, **"Choose a file"**, wildcard=wildcard, style=wx.FD\_OPEN | wx.FD\_FILE\_MUST\_EXIST)  
  
 **if** dialog.ShowModal() == wx.ID\_OK:  
 file\_path = dialog.GetPath()  
 file\_size = os.path.getsize(file\_path)  
 *# if file size is too big* **if** file\_size > 5 \* 1024 \* 1024 \* 1024:  
 wx.MessageBox(**"File size exceeds the maximum allowed (5GB). Please choose a smaller file."**, **"Error"**,  
 wx.OK | wx.ICON\_ERROR)  
 **else**:  
 *# let the logic know upload is requested* self.queue.put((**"upload"**, file\_path))  
 self.upload\_button.Enable(**False**)  
  
 dialog.Destroy()  
  
 **def** on\_download(self, event):  
 *"""  
 after a click on the download start the download  
 """* selected\_index = self.file\_list\_ctrl.GetFirstSelected()  
 **if** selected\_index != -1:  
 selected\_file = self.file\_list[selected\_index][0]  
 *# if file exists in the nitur file* **if** os.path.exists(**f"{**settingCli.NITUR\_FOLDER**}\\{**selected\_file**}"**):  
 result = wx.MessageBox(**"already have this file, copy file to downloads ?"**,  
 **"Error"**,  
 wx.YES\_NO | wx.ICON\_WARNING)  
 *# if the user want to copy* **if** result == wx.YES:  
 shutil.copy2(**f"{**settingCli.NITUR\_FOLDER**}\\{**selected\_file**}"**,  
 **f"{**settingCli.PATH\_TO\_SAVE\_FILES**}\\{**selected\_file**}"**)  
 *# if the user choose dirctory* **elif** os.path.isdir(selected\_file):  
 wx.MessageBox(**"cant download directory, compress to zip file"**, **"Error"**, wx.OK | wx.ICON\_ERROR)  
 **else**:  
 *# let the logic know download is requested* self.queue.put((**"download"**, selected\_file))  
 self.upload\_button.Enable(**False**)  
 self.download\_button.Enable(**False**)  
 **else**:  
 wx.MessageBox(**"Please select a file to download."**, **"Error"**, wx.OK | wx.ICON\_ERROR)  
  
 **def** open\_progress(self, total\_parts, name\_of\_file):  
 *"""  
 opens the progress file* **:param** *total\_parts:number of total parts* **:param** *name\_of\_file: the name of the file* **:return***:  
 """* **if** self.progress\_dialog **is None**:  
 self.total\_parts = total\_parts + 1  
  
 self.progress\_dialog = wx.ProgressDialog(**"Downloading"**, **"Downloading file..."**, maximum=self.total\_parts,  
 parent=self, style=wx.PD\_AUTO\_HIDE | wx.PD\_CAN\_ABORT)  
 *# all the parts and the full hash* self.name\_of\_file = name\_of\_file  
  
 **def** close\_progress(self):  
 *"""  
 closing the progress and checks if the download fail or not* **:return***:  
 """* **if** self.progress\_dialog **is not None**:  
 **if** self.total\_parts\_download >= self.total\_parts:  
 *# Show a message box when the download is complete* wx.MessageBox(**f"Download of {**self.name\_of\_file**} complete!"**, **"Success"**, wx.OK | wx.ICON\_INFORMATION,  
 self)  
 **else**:  
 wx.MessageBox(**f"Download of {**self.name\_of\_file**} fail!"**, **"Fail"**, wx.OK | wx.ICON\_INFORMATION,  
 self)  
 self.resetdownload()  
  
 **def** show\_download\_progress(self):  
 *"""  
 update the progress after a part was downloaded and checks if abort was clicked  
 """  
 # if the progress dialog if open* **if** self.progress\_dialog **is not None**:  
 self.total\_parts\_download += 1  
 abort = self.progress\_dialog.Update(self.total\_parts\_download)  
 *# If download is complete, destroy the progress dialog and rest the progress screen* **if not** abort[0]:  
 **if not** abort[0]:  
 self.queue.put((**"abort"**, self.name\_of\_file))  
 wx.MessageBox(**f"download of {**self.name\_of\_file**} was cancel "**,  
 **"Error"**,  
 wx.OK | wx.ICON\_WARNING)  
 self.resetdownload()  
  
 **def** update\_file\_list(self, new\_file\_list):  
 *"""  
 add the item to the table* **:param** *new\_file\_list: the list of files  
 """* self.file\_list = new\_file\_list  
 self.file\_list\_ctrl.DeleteAllItems()  
 **if** self.file\_list != [[]]:  
 **for** i, file\_info **in** enumerate(self.file\_list):  
 self.file\_list\_ctrl.InsertItem(i, file\_info[0])  
 self.file\_list\_ctrl.SetItem(i, 1, **f"{**naturalsize(file\_info[1])**}"**.lower())  
 self.file\_list\_ctrl.SetItem(i, 2, file\_info[2])  
 self.file\_list\_ctrl.SetItem(i, 3, **f"{**naturaldelta((int(file\_info[1]) / settingCli.BYTES\_PER\_SECOND / int(file\_info[2])) + settingCli.time\_to\_connect\_to\_p2p)**}"**)  
  
 **def** resetdownload(self):  
 *"""  
 restart all the vars* **:return***:  
 """* **if** self.progress\_dialog **is not None**:  
 self.progress\_dialog.Destroy()  
 self.progress\_dialog = **None** self.upload\_button.Enable(**True**)  
 self.download\_button.Enable(**True**)  
 self.total\_parts\_download = 0  
 self.total\_parts = 0  
 self.name\_of\_file = **""**

**class** clientProtocol(object):  
 @staticmethod  
 **def** unpack(data):  
 *"""* **:param** *data: the data from the comm* **:return***: opcode and list of params  
 """* params = data.split(**"$%$"**)  
 opcode = params[0]  
 params.remove(params[0])  
 **return** opcode, params  
  
 @staticmethod  
 **def** unpack\_file(data):  
 *"""* **:param** *data: the data from the comm* **:return***: the opcode and params  
 """* params = data.split(**"$%$"**)  
 opcode = params[1]  
 params.remove(params[1])  
 **return** opcode, params  
  
 @staticmethod  
 **def** build\_part\_of\_key(A):  
 *"""* **:param** *A: part of the key* **:return***: the format  
 """* **return f"00{**A**}"** @staticmethod  
 **def** request\_torrent\_file(file\_name):  
 *"""* **:param** *file\_name: the file name* **:return***: string using the format  
 """* **return f"01$%${**file\_name**}"** @staticmethod  
 **def** request\_upload(path):  
 *"""* **:param** *path:path of the file* **:return***:sting using the format  
 """* **return f"02$%${**path**}"** @staticmethod  
 **def** upload\_file(filename):  
 *"""* **:param** *filename: the fuile name* **:return***: sting using the format  
 """* **return f"$%$01$%${**filename**}"** @staticmethod  
 **def** request\_part\_file(file\_name, number\_of\_part, len\_of\_part):  
 *"""* **:param** *file\_name: the file name* **:param** *number\_of\_part: the number of the part* **:param** *len\_of\_part: the len of each part* **:return***: string using the format  
 """* **return f"01$%${**number\_of\_part**}$%${**file\_name**}$%${**len\_of\_part**}"** @staticmethod  
 **def** send\_file\_part(file\_name, number\_of\_part):  
 *"""* **:param** *file\_name: the file name* **:param** *number\_of\_part: the number of the part* **:return***: string using format  
 """* **return f"01$%${**number\_of\_part**}$%${**file\_name**}$%$"** @staticmethod  
 **def** added\_file\_nitur(file\_name):  
 *"""* **:param** *file\_name: file name* **:return***: string using format  
 """* **return f"01$%${**file\_name**}"** @staticmethod  
 **def** removed\_file\_nitur(file\_name):  
 *"""* **:param** *file\_name: file name* **:return***: string using format  
 """* **return f"02$%${**file\_name**}"** @staticmethod  
 **def** changed\_file\_nitur(file\_name):  
 *"""* **:param** *file\_name: file name* **:return***: string using format  
 """* **return f"03$%${**file\_name**}"** @staticmethod  
 **def** changed\_file\_name\_new\_nitur(file\_name):  
 *"""* **:param** *file\_name: file name* **:return***: sting using format  
 """* **return f"05$%${**file\_name**}"**

**import** os  
**from** allcode **import** settingSer  
  
  
**class** client\_files(object):  
 @staticmethod  
 **def** get\_part\_of\_file(path\_of\_file, number\_of\_part):  
 *"""* **:param** *path\_of\_file: path of the file* **:param** *number\_of\_part: number of the part* **:return***: the data of the the part  
 """* **with** open(**fr"{**path\_of\_file**}"**, **'rb'**) **as** f:  
 **if** number\_of\_part == -1:  
 data = f.read()  
 **else**:  
 f.seek(number\_of\_part \* settingSer.len\_of\_part\_of\_file)  
 data = f.read(settingSer.len\_of\_part\_of\_file)  
 **return** data  
  
 @staticmethod  
 **def** save\_file(nitur\_path, name\_of\_file, data):  
 *"""  
 save the file* **:param** *nitur\_path: the path to save* **:param** *name\_of\_file:the name of the file* **:param** *data:the data of the file* **:return***:  
 """* **with** open(**f"{**nitur\_path**}\\{**name\_of\_file**}"**, **'wb'**) **as** f:  
 f.write(data)  
  
 @staticmethod  
 **def** delete\_file(path\_of\_file):  
 *"""  
 delete the file* **:param** *path\_of\_file:  
 """* **if** os.path.exists(**fr"{**path\_of\_file**}"**):  
 **try**:  
 os.remove(**fr"{**path\_of\_file**}"**)  
 **except** Exception **as** e:  
 print(e, **"file is open"**)

**import** threading  
**import** socket  
**from** allcode **import** Encryption\_Decryption  
**import** sys  
**from** clientCode **import** clientProtocol  
  
  
**class** Clientcomm(object):  
 **def** \_\_init\_\_(self, server\_ip, message\_queue, port, zfill\_number, timer=1000):  
 *"""* **:param** *server\_ip: ip of the server* **:param** *message\_queue: the queue to put the messages in for the logic* **:param** *port:the server port* **:param** *zfill\_number: the number to zfill the message  
 """* self.server\_ip = server\_ip  
 self.message\_queue = message\_queue  
 self.port = port  
 self.zfill\_number = zfill\_number  
 self.client\_socket = **None** self.crypt\_object = **None** self.is\_socket\_open = **True** self.timer = timer  
 threading.Thread(target=self.\_recv\_messages).start()  
  
 **def** \_recv\_messages(self):  
 *"""  
 the main loop the recv the messages  
 """* self.client\_socket = socket.socket()  
 **try**:  
 self.client\_socket.connect((self.server\_ip, self.port))  
 **except** Exception **as** e:  
 print(e, **f"cant connect to {**self.server\_ip**}"**)  
 **else**:  
 self.\_xchange\_key()  
 **while** self.is\_socket\_open:  
 **try**:  
 self.client\_socket.settimeout(self.timer)  
 len\_of\_message = self.client\_socket.recv(self.zfill\_number).decode()  
 **except** socket.timeout:  
 print(**"timeout"**, self.server\_ip)  
 **continue  
 except** Exception **as** e:  
 print(e)  
 sys.exit()  
 **if** len\_of\_message == **''**:  
 self.close\_socket()  
 **break  
 try**:  
 self.client\_socket.settimeout(self.timer)  
 encrypt\_message = self.client\_socket.recv(int(len\_of\_message)).decode()  
 **except** socket.timeout:  
 **continue  
 except** Exception **as** e:  
 print(e)  
 sys.exit()  
 message = self.crypt\_object.decrypt(encrypt\_message).decode()  
 opcode, params = clientProtocol.clientProtocol.unpack(message)  
 *# if a header was recved* **if** opcode == **"01"**:  
 self.\_recv\_file(params)  
 **else**:  
 self.message\_queue.put(message)  
  
 **def** \_recv\_file(self, params):  
 *"""* **:param** *params: list of params (the header)* **:return***: put the data and the header of the file in the queue  
 """* number\_of\_part, file\_name, data\_len = int(params[0]), params[1], int(params[2])  
 data\_part = bytearray()  
 **while** data\_len >= 1024:  
 **try**:  
 self.client\_socket.settimeout(self.timer)  
 message = self.client\_socket.recv(1024)  
 data\_part.extend(message)  
 **except** socket.timeout:  
 data\_part = -1  
 print(**"timeout"**)  
 data\_len = 0  
 **break  
 except** Exception **as** e:  
 print(e)  
 self.close\_socket()  
 sys.exit()  
 **else**:  
 data\_len -= len(message)  
 **if** data\_len != 0:  
 **try**:  
 self.client\_socket.settimeout(self.timer)  
 data\_part += self.client\_socket.recv(data\_len)  
 **except** socket.timeout:  
 data\_part = -1  
 print(**"timeout"**)  
 **except** Exception **as** e:  
 print(e)  
 self.close\_socket()  
 sys.exit()  
 data\_part = self.crypt\_object.decrypt(data\_part)  
 self.message\_queue.put((self.server\_ip, file\_name, number\_of\_part, data\_part))  
  
 **def** \_xchange\_key(self):  
 *"""  
 create the AES object  
 """* b, B = Encryption\_Decryption.AES\_encryption.get\_dif\_Num()  
 message = clientProtocol.clientProtocol.build\_part\_of\_key(B)  
 len\_of\_message = str(len(message)).zfill(self.zfill\_number)  
 **try**:  
 self.client\_socket.send(**f"{**len\_of\_message**}{**message**}"**.encode())  
 **except** Exception **as** e:  
 print(e)  
 sys.exit()  
 **try**:  
 len\_of\_message = self.client\_socket.recv(self.zfill\_number).decode()  
 message = self.client\_socket.recv(int(len\_of\_message)).decode()  
 **except** Exception **as** e:  
 print(e)  
 sys.exit()  
 **if** message[:2] == **"00"**:  
 A = int(message[2:])  
 crypt\_object = Encryption\_Decryption.AES\_encryption.set\_key(A, b)  
 *# exchange keys and create cryptObject* self.crypt\_object = crypt\_object  
  
 **def** send(self, message):  
 *"""* **:param** *message: the message to send  
 send the message after encrypt to the server  
 """* **while** self.crypt\_object **is None**:  
 **continue  
 if** self.crypt\_object **is not None and** self.is\_socket\_open:  
 encrypt\_msg = self.crypt\_object.encrypt(message)  
 len\_encrypt\_msg = str(len(encrypt\_msg)).zfill(self.zfill\_number).encode()  
 **try**:  
 self.client\_socket.send(len\_encrypt\_msg + encrypt\_msg)  
 **except** Exception **as** e:  
 print(e)  
  
 **def** send\_file(self, data, header):  
 *"""  
 send the file to the server* **:param** *data: the data of the file* **:param** *header: the header of the file  
 """* **while** self.crypt\_object **is None**:  
 **continue  
 if** self.crypt\_object **is not None and** self.is\_socket\_open:  
 encrypt\_data = self.crypt\_object.encrypt(data)  
 len\_encrypt\_data = str(len(encrypt\_data)).zfill(self.zfill\_number).encode()  
 encrypt\_header = self.crypt\_object.encrypt(len\_encrypt\_data + header.encode())  
 len\_encrypt\_header = str(len(encrypt\_header)).zfill(self.zfill\_number).encode()  
 self.client\_socket.send(len\_encrypt\_header + encrypt\_header + encrypt\_data)  
  
 **def** close\_socket(self):  
 *"""  
 close the socket and the main loop  
 """* self.is\_socket\_open = **False** self.client\_socket.close()

GENERAL\_PORT = 1500  
NITUR\_PORT = 1600  
PATH\_OF\_TORRENTS\_FOLDER = **f"D:\\torrent\_files"**len\_of\_part\_of\_file = 512\*1024

SERVER\_IP = **'192.168.4.96'**GENERAL\_PORT = 1500  
NITUR\_PORT = 1600  
P2P\_PORT = 3333  
BYTES\_PER\_SECOND = 19000000  
*# time\_to\_connect = 0.5*PATH\_TO\_SAVE\_FILES = **fr"C:**\Users**\talmid\Downloads"**NITUR\_FOLDER = **fr"D:\.nitur"**IMAGES\_PATH = **fr"T:\public\יב\imri\nexus-main\\"**time\_to\_connect\_to\_p2p = 0.5

**class** serverProtocol(object):  
 @staticmethod  
 **def** unpack(data):  
 *"""* **:param** *data: the data from the comm* **:return***: opcode and params as list  
 """* params = data.split(**"$%$"**)  
 opcode = params[0]  
 params.remove(params[0])  
 **return** opcode, params  
  
 @staticmethod  
 **def** unpack\_file(data):  
 *"""* **:param** *data:the data from the comm* **:return***:opcode and params as list  
 """* params = data.split(**"$%$"**)  
 opcode = params[1]  
 params.remove(params[1])  
 **return** opcode, params  
  
 @staticmethod  
 **def** build\_part\_of\_key(A):  
 *"""* **:param** *A: part of the key* **:return***: a with format  
 """* **return f"00{**A**}"** @staticmethod  
 **def** Response\_for\_torrent\_request(file\_name):  
 *"""* **:param** *file\_name: name of the file* **:return***: response for the torrent request  
 """* **return f"01$%$0$%${**file\_name**}$%$"** @staticmethod  
 **def** Response\_for\_upload\_request(path\_of\_file, port):  
 *"""* **:param** *path\_of\_file: path* **:param** *port: port to use* **:return***: path and port with format  
 """* **return f"02$%${**port**}$%${**path\_of\_file**}"** @staticmethod  
 **def** Response\_for\_upload():  
 **return "111"** @staticmethod  
 **def** Delete\_file\_from\_folder(file\_name):  
 *"""* **:param** *file\_name: name of file to delete* **:return***: the name with format  
 """* **return f"03$%${**file\_name**}"** @staticmethod  
 **def** Create\_string\_of\_list(list\_of\_open\_file):  
 *"""* **:param** *list\_of\_open\_file: list of all the open file* **:return***: a sting in format with all the data on the files  
 """* message = **"03"  
 for** file **in** list\_of\_open\_file:  
 **if** list\_of\_open\_file[file][1] != 0:  
 message += **f"$%${**file**},{**list\_of\_open\_file[file][0]**},{**list\_of\_open\_file[file][1]**}"  
 return** message

**import** socket  
**import** select  
**import** threading  
**from** allcode **import** Encryption\_Decryption  
**from** allcode **import** settingSer  
**from** allcode **import** serverProtocol  
**from** allcode **import** settingCli  
  
  
**class** ServerComm(object):  
 **def** \_\_init\_\_(self, port, message\_queue, zfill\_number):  
 *"""* **:param** *port: the port of the server* **:param** *message\_queue: the queue that recv the queue* **:param** *zfill\_number: zill number to use in the zfill  
 """* self.port = port  
 self.message\_queue = message\_queue  
 self.zfill\_number = zfill\_number  
 self.open\_clients = {}  
 self.server\_socket = **None** self.is\_socket\_open = **True** threading.Thread(target=self.\_recv\_messages).start()  
  
 **def** \_recv\_messages(self):  
 *"""  
 the main loop that recv the messages* **:return***: None  
 """* self.server\_socket = socket.socket()  
 self.server\_socket.bind((**"0.0.0.0"**, self.port))  
 self.server\_socket.listen(3)  
 **while** self.is\_socket\_open:  
 rlist, wlist, xlist = select.select([self.server\_socket] + list(self.open\_clients.keys()),  
 list(self.open\_clients.keys()), [], 0.3)  
 **for** current\_socket **in** rlist:  
 *# new client* **if** current\_socket **is** self.server\_socket:  
 (new\_client, addr) = self.server\_socket.accept()  
 threading.Thread(target=self.\_xchange\_key, args=(new\_client, addr)).start()  
 **else**:  
 **try**:  
 len\_of\_message = int(current\_socket.recv(self.zfill\_number).decode())  
 encrypt\_message = current\_socket.recv(len\_of\_message)  
 **except** Exception **as** e:  
 print(e)  
 *# if disconnect let the logic know* **if** self.port == settingSer.NITUR\_PORT:  
 self.message\_queue.put((self.open\_clients[current\_socket][0], **"-1$%$"**))  
 **del** self.open\_clients[current\_socket]  
 **else**:  
 message = self.open\_clients[current\_socket][1].decrypt(encrypt\_message).decode()  
 *# if the message is string* **if** self.port **in** [settingSer.GENERAL\_PORT, settingSer.NITUR\_PORT, settingCli.P2P\_PORT]:  
 self.message\_queue.put((self.open\_clients[current\_socket][0], message))  
 *# the message is file* **else**:  
 self.\_recv\_file(current\_socket, message)  
  
 **def** \_xchange\_key(self, new\_client, addr):  
 *"""  
 get the b from the client* **:return***: create a AES object  
 """* a, A = Encryption\_Decryption.AES\_encryption.get\_dif\_Num()  
 **try**:  
 len\_of\_message = new\_client.recv(self.zfill\_number).decode()  
 message = new\_client.recv(int(len\_of\_message)).decode()  
 **except** Exception **as** e:  
 print(e)  
 **else**:  
 B = int(message[2:])  
 **if** message[:2] == **"00"**:  
 message = serverProtocol.serverProtocol.build\_part\_of\_key(A)  
 len\_of\_message = str(len(message)).zfill(self.zfill\_number)  
 **try**:  
 new\_client.send(**f"{**len\_of\_message**}{**message**}"**.encode())  
 **except** Exception **as** e:  
 print(e)  
 **else**:  
 *# create cryptObject* cryptobject = Encryption\_Decryption.AES\_encryption.set\_key(B, a)  
 self.open\_clients[new\_client] = [addr[0], cryptobject]  
 **if** self.port == settingSer.GENERAL\_PORT:  
 self.message\_queue.put((self.open\_clients[new\_client][0], **"03"**))  
  
 **def** \_find\_socket\_by\_ip(self, find\_ip):  
 *"""* **:param** *find\_ip: the ip* **:return***: the socket  
 """* **for** key, value **in** self.open\_clients.items():  
 **if** value[0] == find\_ip:  
 **return** key  
  
 **def** send(self, message, ip):  
 *"""* **:param** *message: the message* **:param** *ip: the ip to send to  
 """* current\_socket = self.\_find\_socket\_by\_ip(ip)  
 *# if the socket is open and current socket is not none* **if** current\_socket **is not None and** self.is\_socket\_open:  
 encrypt\_msg = self.open\_clients[current\_socket][1].encrypt(message.encode())  
 len\_encrypt\_msg = str(len(encrypt\_msg)).zfill(self.zfill\_number).encode()  
 **try**:  
 current\_socket.send(len\_encrypt\_msg+encrypt\_msg)  
 **except** Exception **as** e:  
 print(e)  
 **del** self.open\_clients[current\_socket]  
  
 **def** send\_file(self, data, header, ip):  
 *"""* **:param** *data: the data of the file* **:param** *header: the header of the file* **:param** *ip: the ip to send to  
 """* **if** self.is\_socket\_open:  
 current\_socket = self.\_find\_socket\_by\_ip(ip)  
 *# if the current socket is not None* **if** current\_socket:  
 crypto = self.open\_clients[current\_socket][1]  
 encrypt\_data = crypto.encrypt(data)  
 len\_encrypt\_data = str(len(encrypt\_data)).zfill(self.zfill\_number).encode()  
 encrypt\_header = crypto.encrypt(header.encode()+ len\_encrypt\_data)  
 len\_encrypt\_header = str(len(encrypt\_header)).zfill(self.zfill\_number).encode()  
 **try**:  
 current\_socket.send(len\_encrypt\_header + encrypt\_header + encrypt\_data)  
 **except** Exception **as** e:  
 print(e)  
 **del** self.open\_clients[current\_socket]  
  
 **def** sendall(self, message):  
 *"""* **:param** *message:the message* **:return***: send to everyone the message  
 """* **for** sock **in** self.open\_clients:  
 **try**:  
 ip = self.open\_clients[sock][0]  
 self.send(message, ip)  
 **except** Exception **as** e:  
 print(e)  
  
 **def** \_recv\_file(self, current\_socket, message):  
 *"""* **:param** *current\_socket: the socket to recv from* **:param** *message: the header  
 """* opcode, params = serverProtocol.serverProtocol.unpack\_file(message)  
 len\_encrypt\_data = int(params[0])  
 full\_data = bytearray()  
 **while** len\_encrypt\_data >= 1024:  
 **try**:  
 data = current\_socket.recv(1024)  
 full\_data.extend(data)  
 **except** Exception **as** e:  
 print(e)  
 **del** self.open\_clients[current\_socket]  
 **else**:  
 len\_encrypt\_data -= len(data)  
 **if** len\_encrypt\_data != 0:  
 **try**:  
 data = current\_socket.recv(len\_encrypt\_data)  
 full\_data.extend(data)  
 **except** Exception **as** e:  
 print(e)  
 **del** self.open\_clients[current\_socket]  
 data = self.open\_clients[current\_socket][1].decrypt(full\_data)  
 params.append(data)  
 params[0] = len(data)  
 self.message\_queue.put((self.open\_clients[current\_socket][0], params))  
  
 **def** close\_socket(self):  
 *"""* **:return***: close the socket  
 """* self.is\_socket\_open = **False**

**import** base64  
**from** Cryptodome.Cipher **import** AES  
**from** Cryptodome **import** Random  
**import** hashlib  
**import** random  
  
  
**class** AES\_encryption(object):  
 **def** \_\_init\_\_(self, key):  
 *"""* **:param** *key: the key  
 """* self.bs = AES.block\_size  
 self.key = hashlib.sha256(str(key).encode()).digest()  
  
 **def** encrypt(self, message):  
 *"""* **:param** *message: the message to encrypt* **:return***: the encrypted message  
 """* **if** type(message) == str:  
 message = message.encode()  
 raw = self.\_pad(message)  
 iv = Random.new().read(AES.block\_size)  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
 **return** base64.b64encode(iv + cipher.encrypt(raw))  
  
 **def** decrypt(self, encrypt\_message):  
 *"""* **:param** *encrypt\_message: the message after encrypt* **:return***: the origin message  
 """* enc = base64.b64decode(encrypt\_message)  
 iv = enc[:AES.block\_size]  
 cipher = AES.new(self.key, AES.MODE\_CBC, iv)  
 **return** self.\_unpad(cipher.decrypt(enc[AES.block\_size:]))  
  
 **def** \_pad(self, message):  
 *"""* **:param** *message: the message* **:return***: message with pading  
 """* **return** message + (self.bs - len(message) % self.bs) \* chr(self.bs - len(message) % self.bs).encode()  
  
 **def** \_unpad(self, message):  
 *"""* **:param** *message: the message with pad* **:return***: the message  
 """* **return** message[:-ord(message[len(message) - 1:])]  
  
 @staticmethod  
 **def** hash(message):  
 *"""* **:param** *message: the message* **:return***: the message after hash  
 """* **return** hashlib.sha256(message).digest()  
  
 @staticmethod  
 **def** get\_dif\_Num():  
 *"""* **:return***: 2 numbers in dif format  
 """* a = random.randint(1, p)  
 **return** a, str((g\*\*a) % p)  
  
 @staticmethod  
 **def** set\_key(B, a):  
 *"""* **:param** *A:* **:param** *B:* **:return***: the key  
 """* **return** AES\_encryption((int(B)\*\*int(a)) % p)  
  
  
p = 7723  
g = 1229